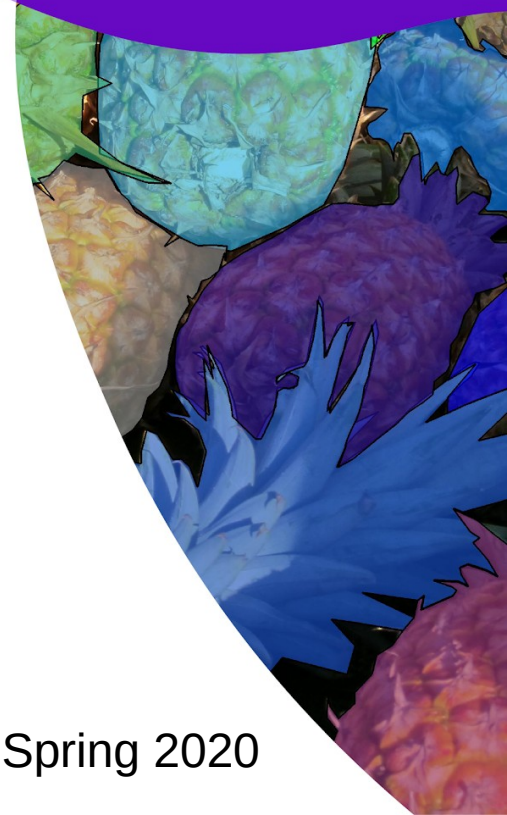
NEW YORK UNIVERSITY

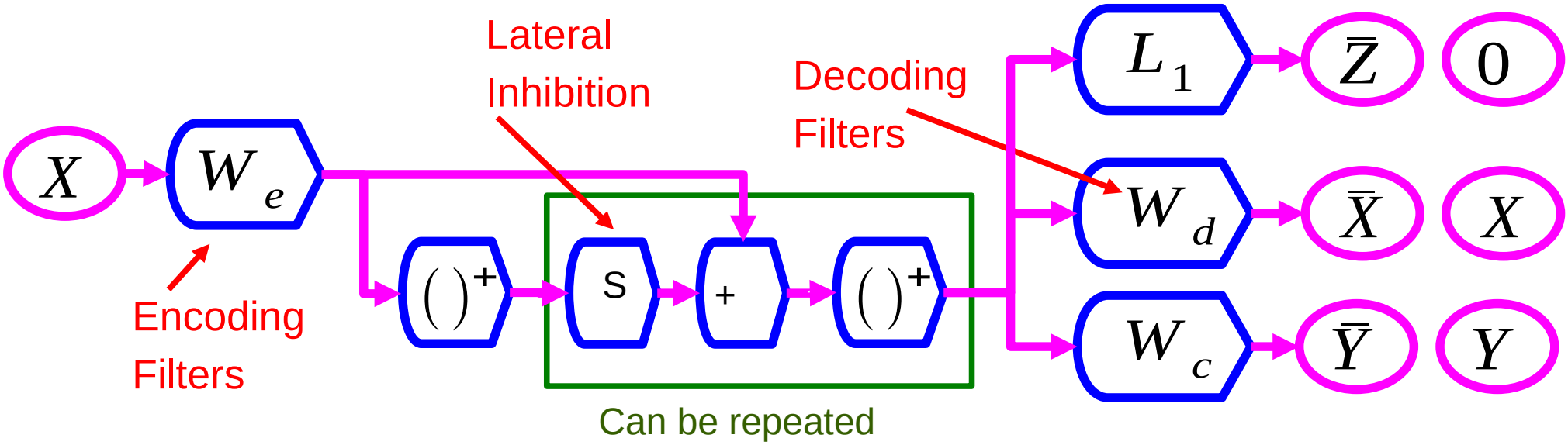
Energy-Based Models (part 3)

http://bit.ly/DLSP20

Yann LeCun
NYU - Courant Institute & Center for Data Science
Facebook AI Research
http://yann.lecun.com
TAs: Alfredo Canziani, Mark Goldstein



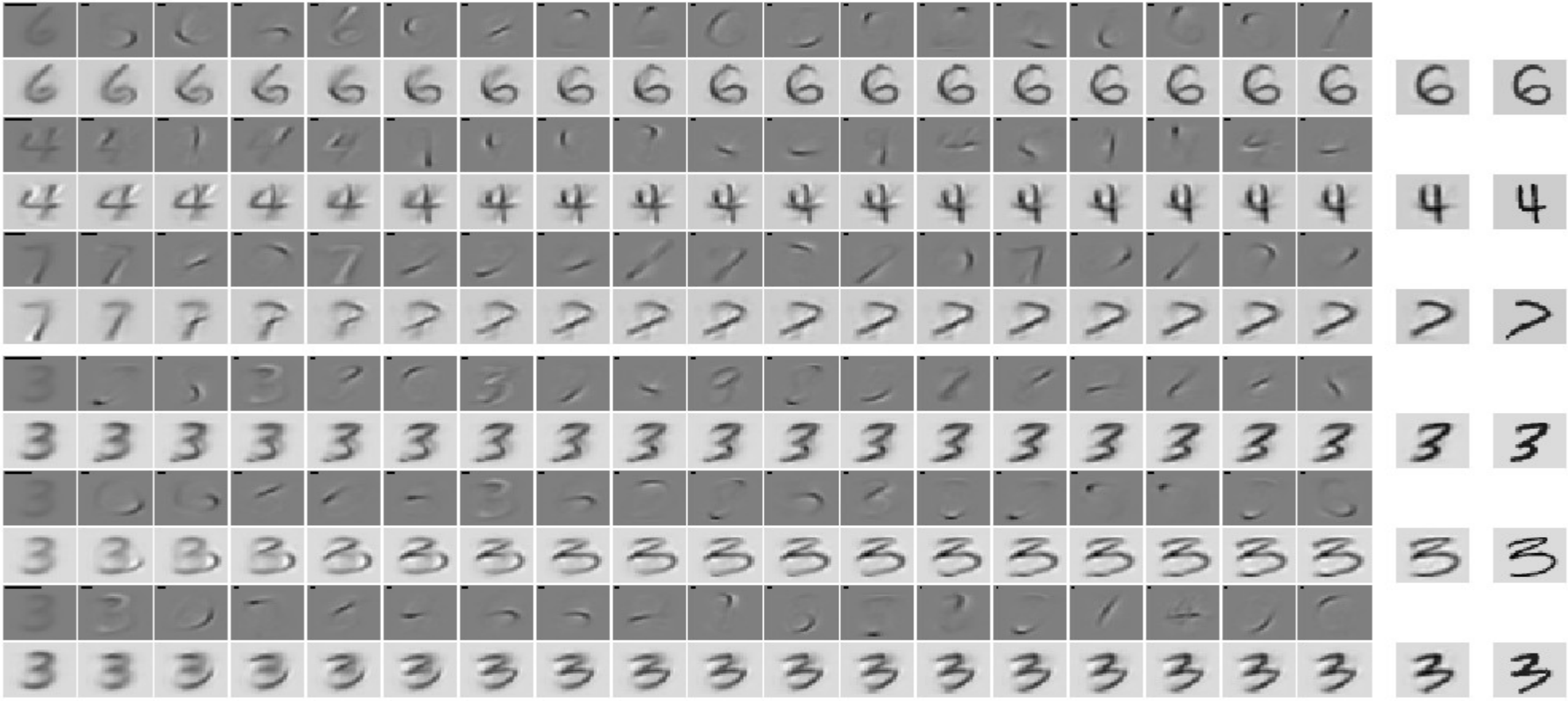
Discriminative Recurrent Sparse Auto-Encoder (DrSAE)



[Rolfe & LeCun ICLR 2013]

DrSAE Discovers manifold structure of handwritten digits

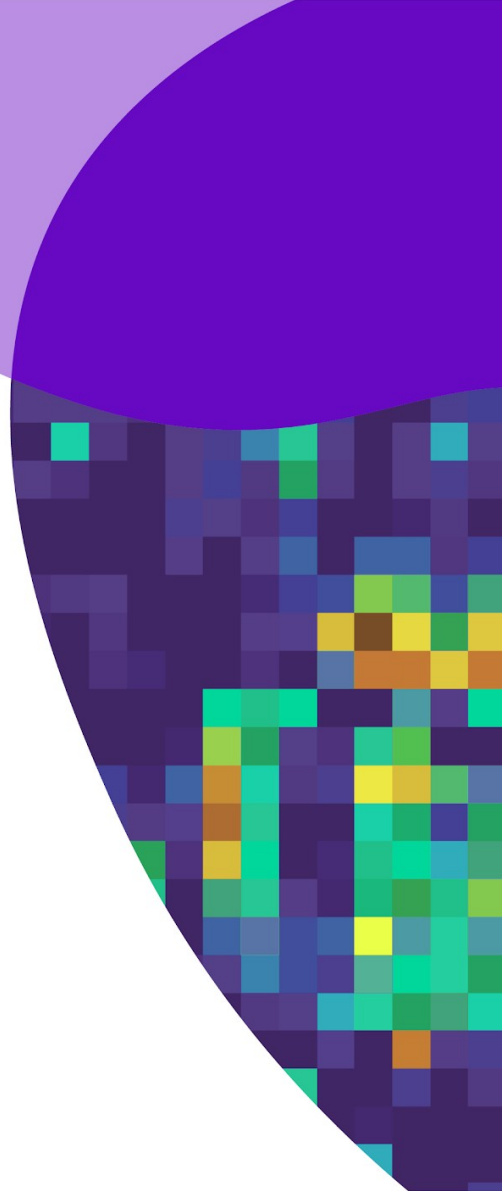
- Image = prototype + sparse sum of “parts” (to move around the



FACEBOOK AI

Group Sparsity, Structured Sparsity

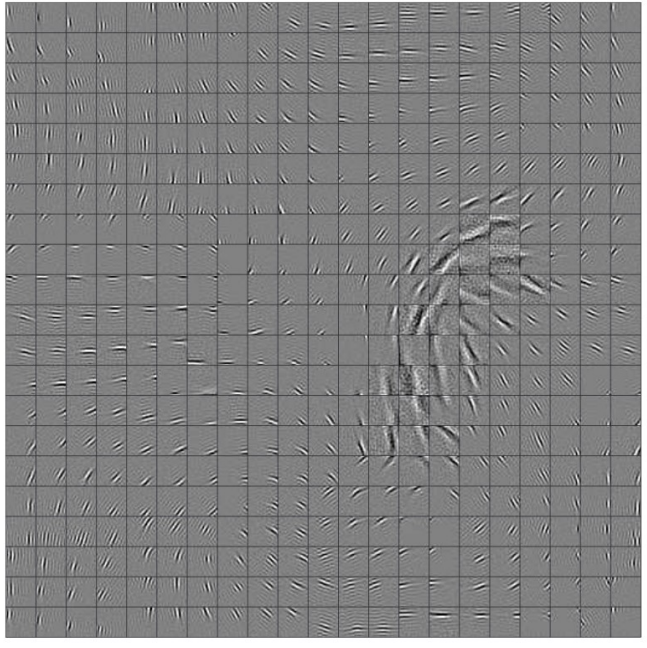
Sparsity after pooling →
invariant features



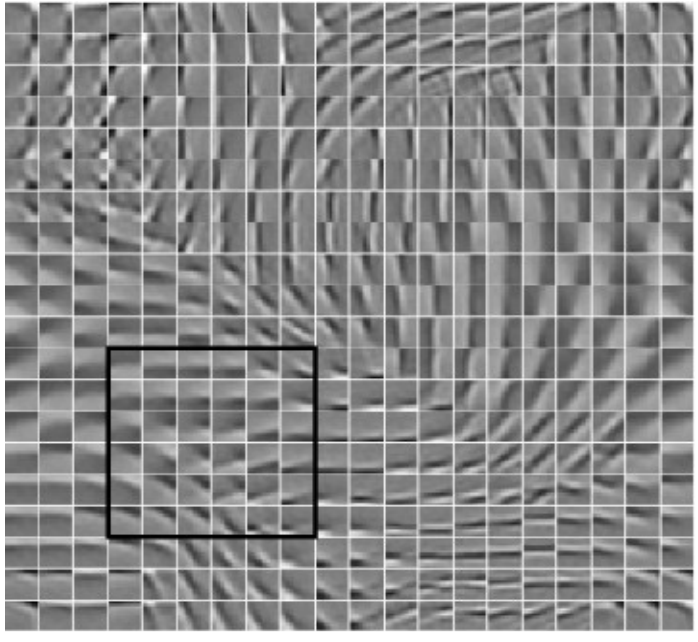
Learning invariant features

- **Sparsity over pooling units → group sparsity**
  - [Hyvarinen & Hoyer 2001], [Osindero et al. Neural Comp. 2006], [Kavukcuoglu et al. CVPR 2009], [Gregor & LeCun arXiv:1006.044], [Mairal et al. 2011].

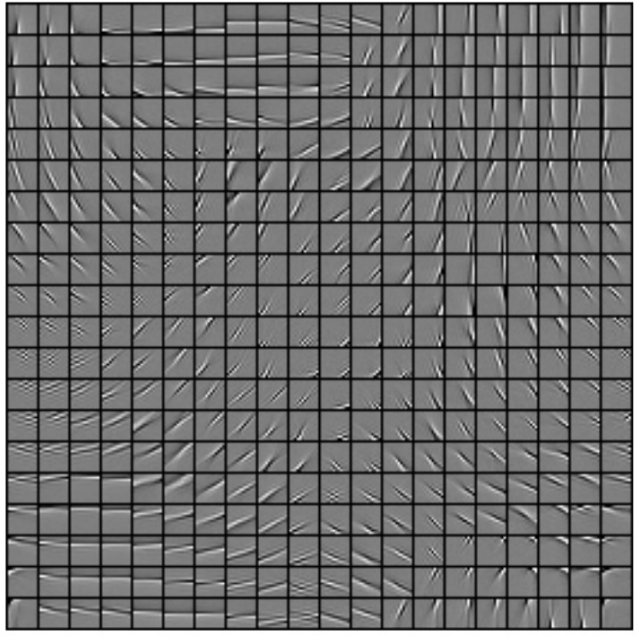
[Osindero 2006]



[Kavukcuoglu 2009]

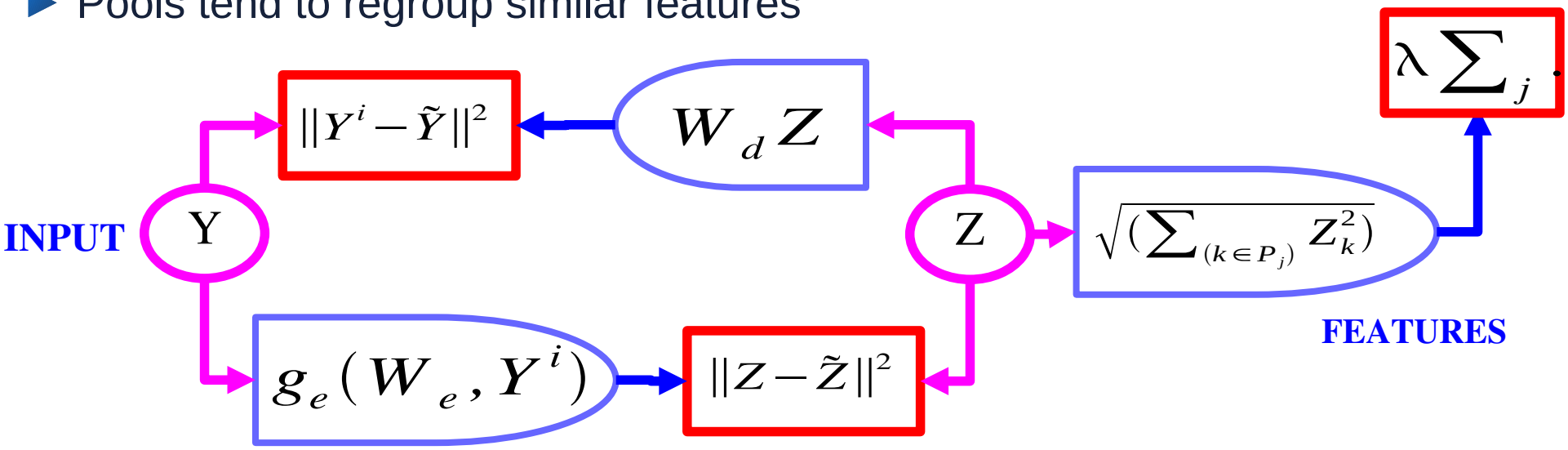


[Mairal 2011]



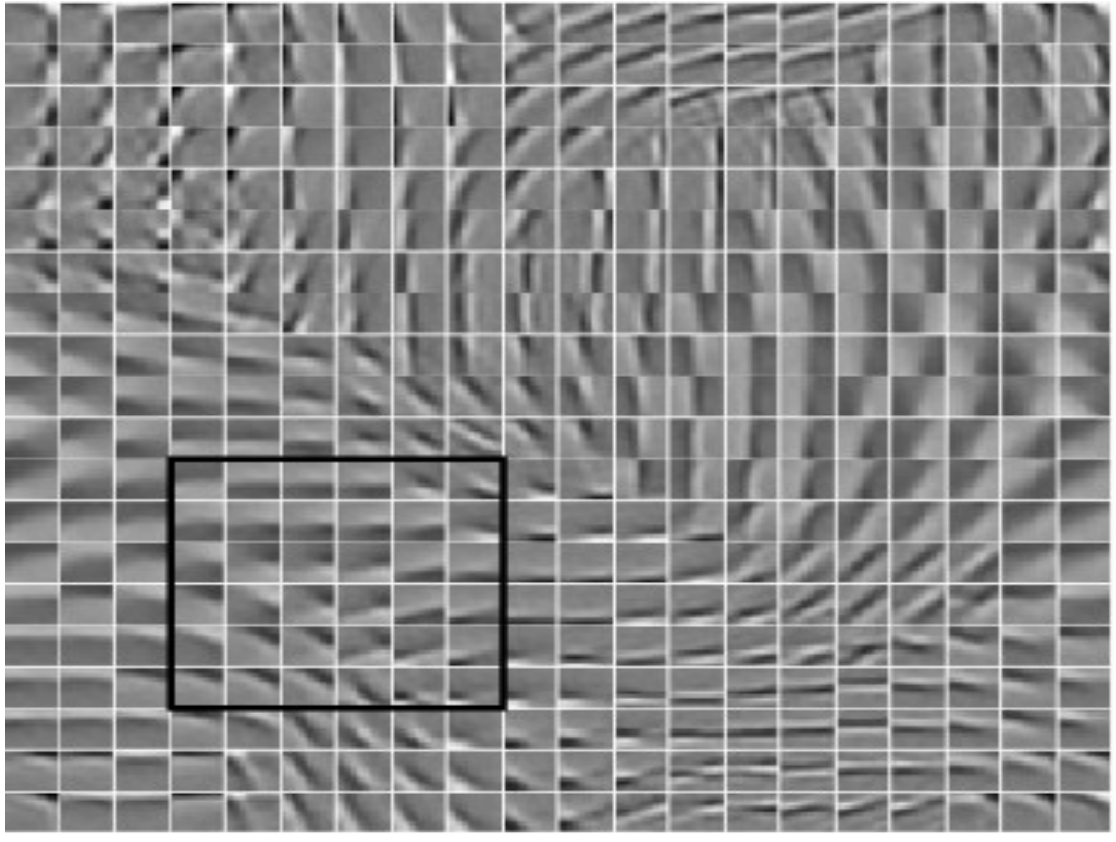
AE with Group Sparsity [Kavukcuoglu et al. CVPR 2009]

- **Trains a sparse AE to produce invariant features**
- **Could we devise a similar method that learns the pooling layer as well?**
- **Group sparsity on pools of features**
  - Minimum number of pools must be non-zero
  - Number of features that are on within a pool doesn't matter
  - Pools tend to regroup similar features



Pooling over features in a topographic map.

- **The pools are 6x6 blocks of features arranged in a 2D torus**
- **While training, the filters arrange themselves spontaneously so that similar filters enter the same pool.**
- **The pooling units can be seen as complex cells**
- **They are invariant to local transformations of the input**
  - For some it's translations, for others rotations, or other transformations.



Pinwheels!

- **The so-called “pinwheel” pattern is observed in the primary visual cortex.**

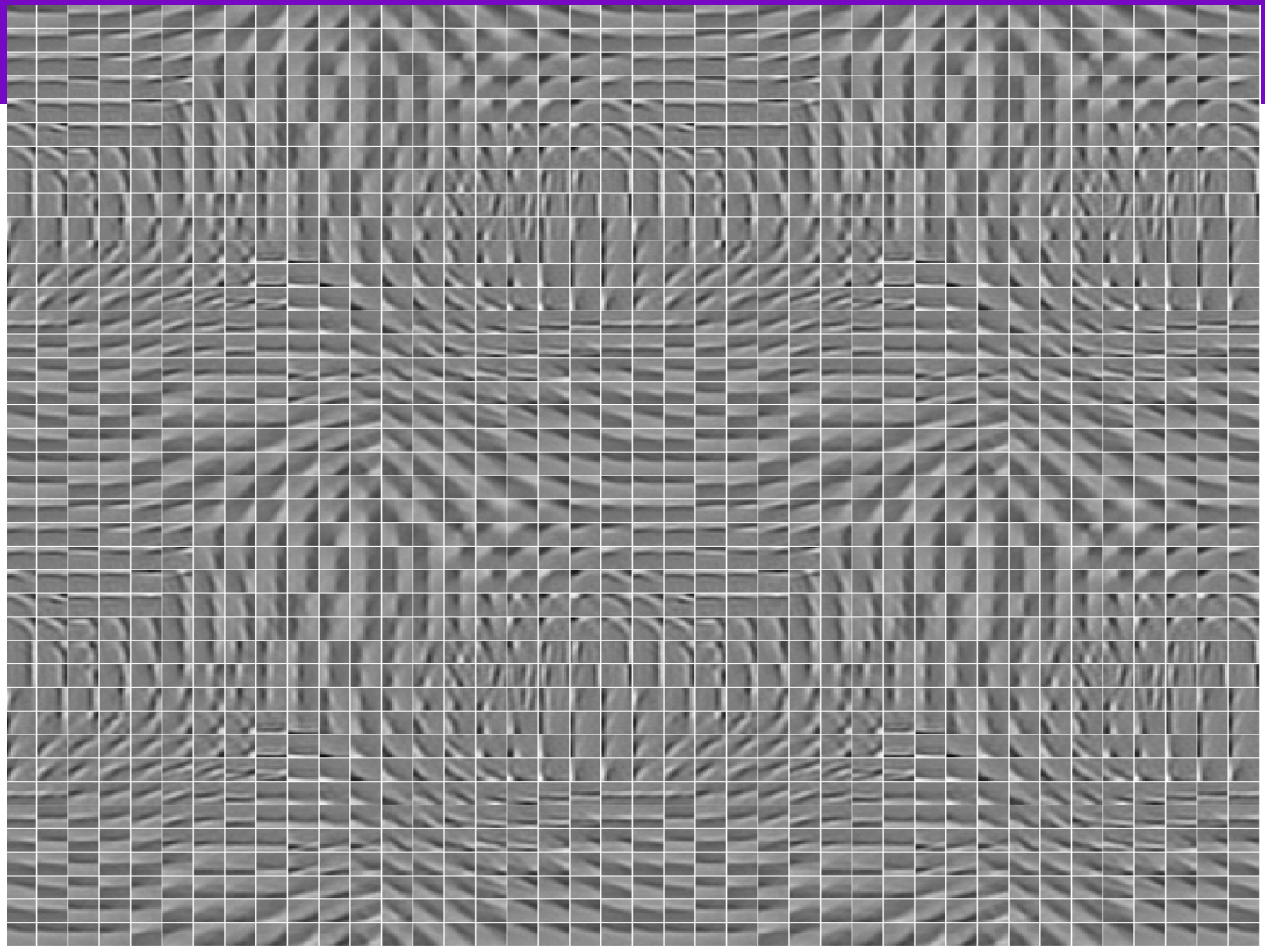


Image-level training, local filters but no weight sharing!

- **Training on 115x115 images. Kernels are 15x15 (not shared across space!)**
- **[Gregor & LeCun arXiv:1006.044]**
  - "Emergence of Complex-Like Cells in a Temporal Product Network with Local Receptive Fields"

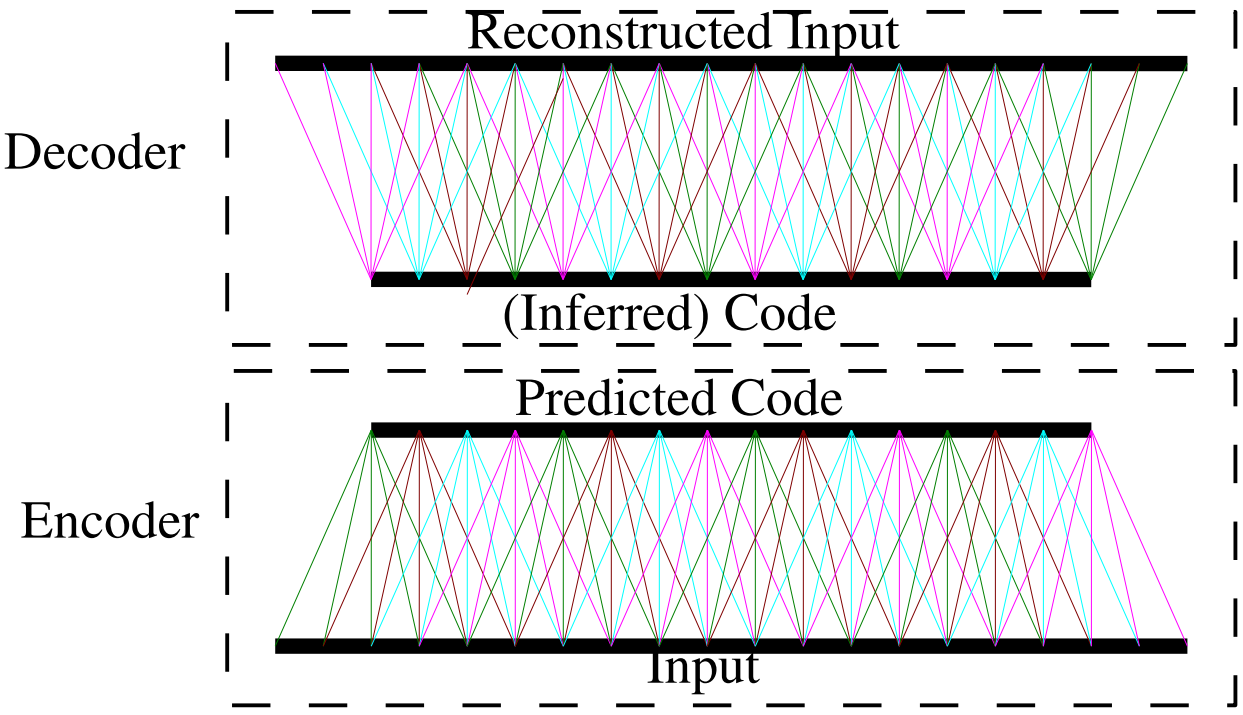


Image-level training, local filters but no weight sharing!

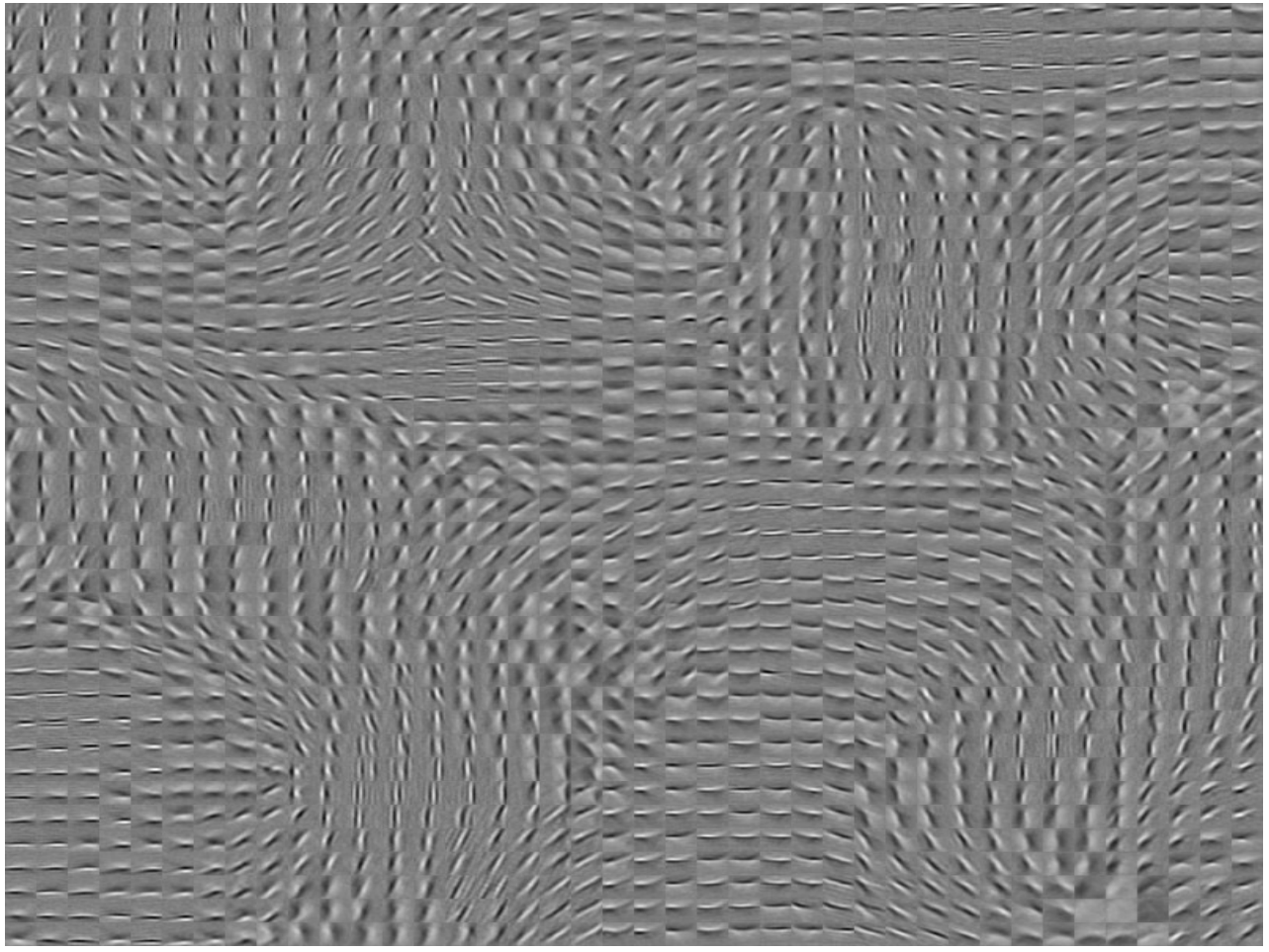
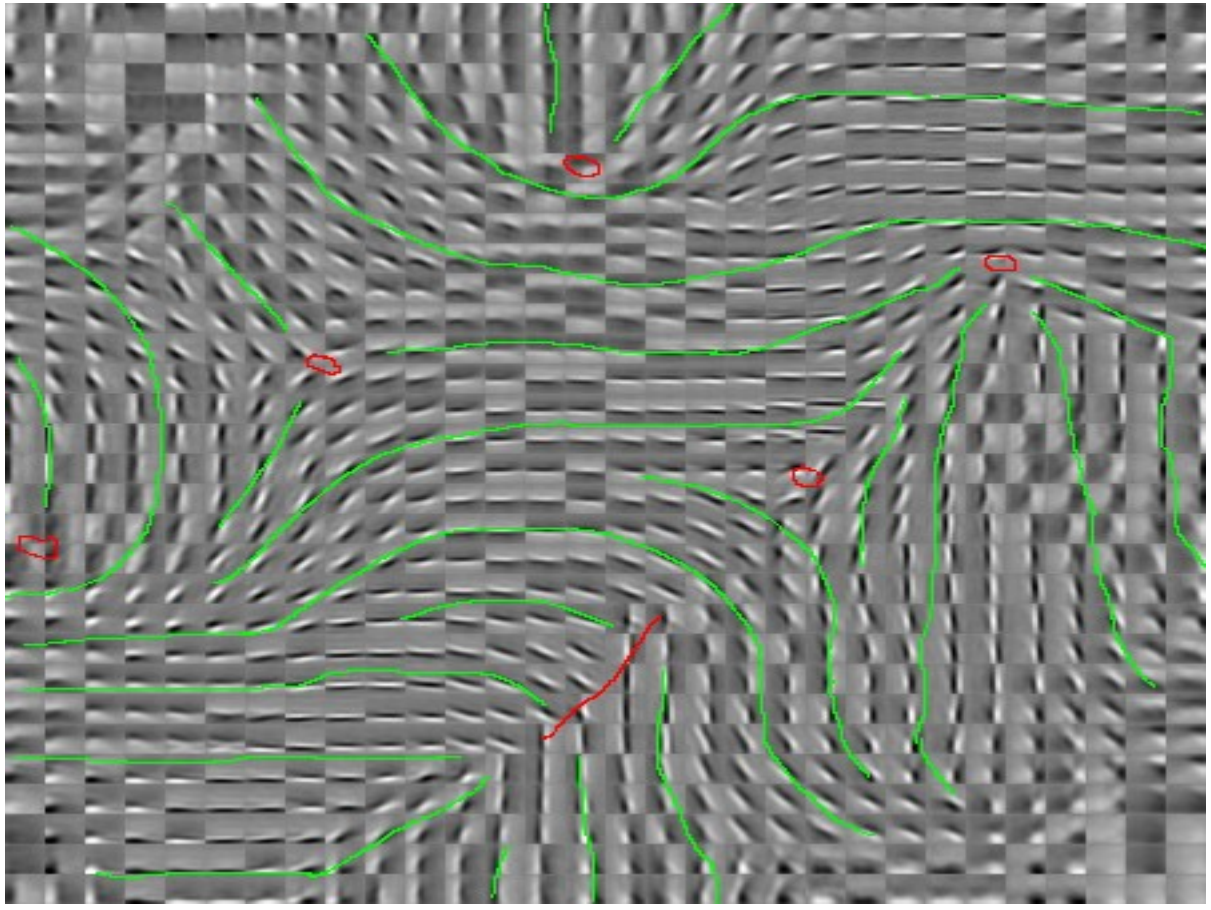


Image-level training, local filters but no weight sharing!

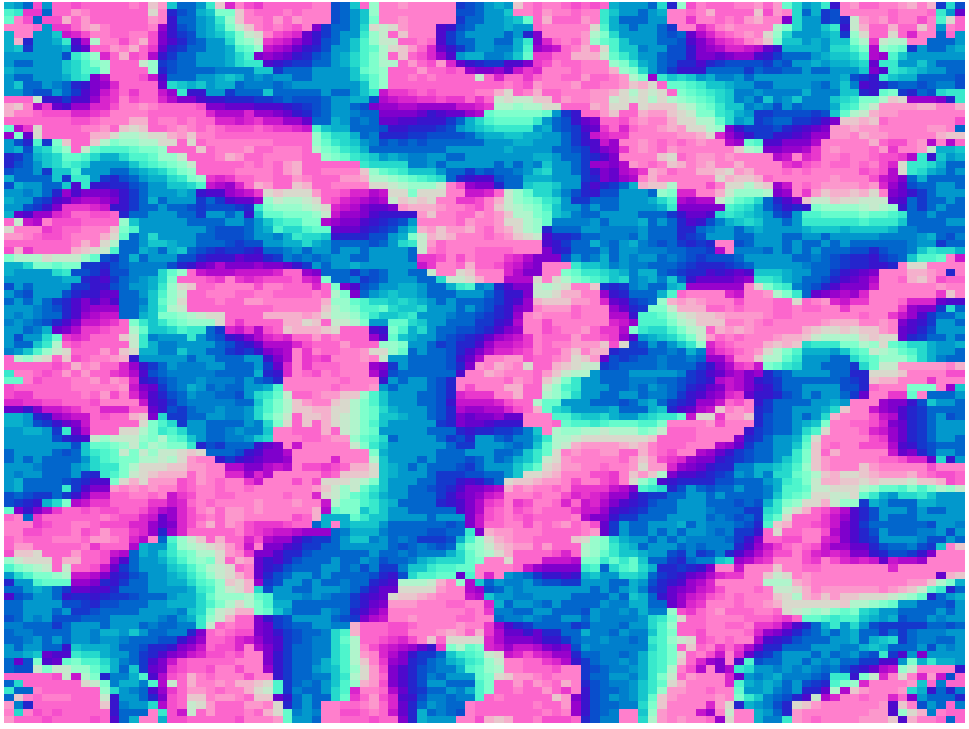
- **Training on 115x115 images. Kernels are 15x15 (not shared across space!)**



Orientation Selectivity Map

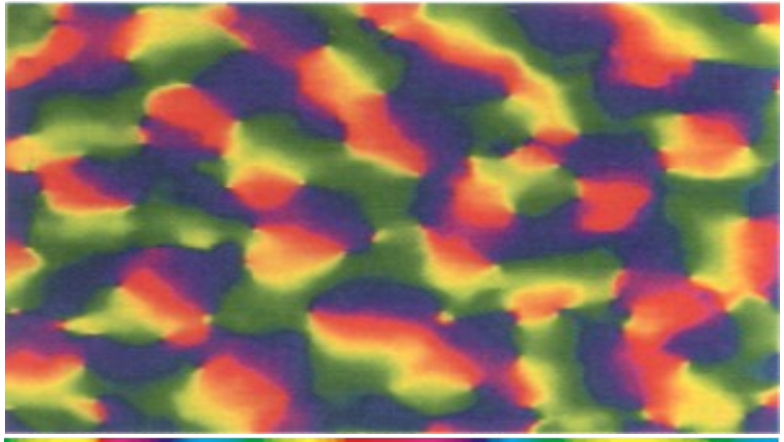
119x119 Image Input, 100x100 Code

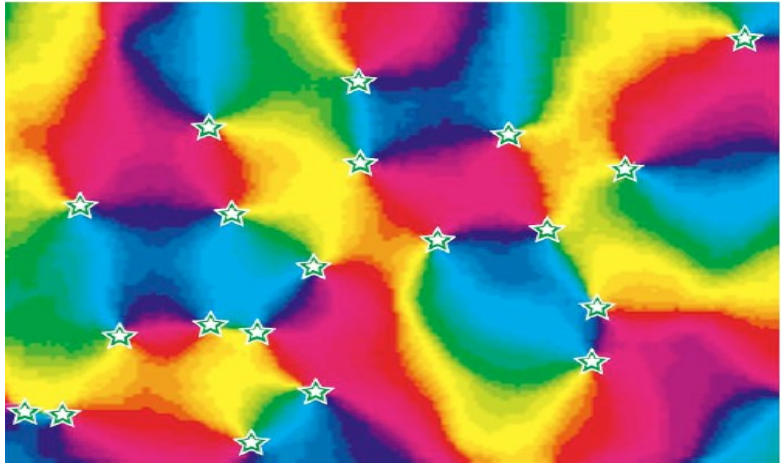
20x20 Receptive field size, sigma=5



Michael C. Crair, et. al. The Journal of Neurophysiology Vol. 77 No. 6 June 1997, pp. 3381-3385 (**Cat**)

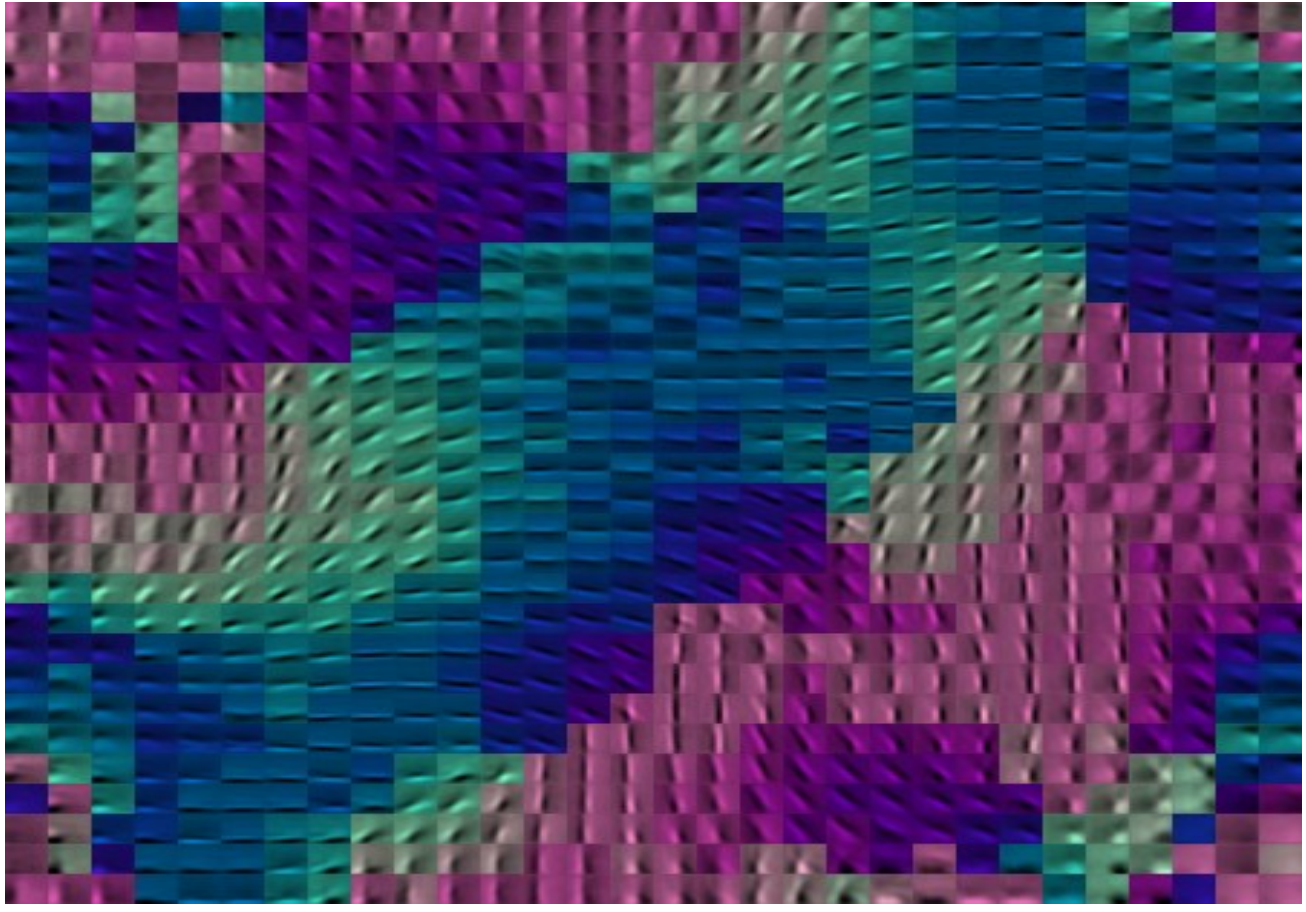
K Obermayer and GG Blasdel, Journal of Neuroscience, Vol 13, 4114-4129 (**Monkey**)





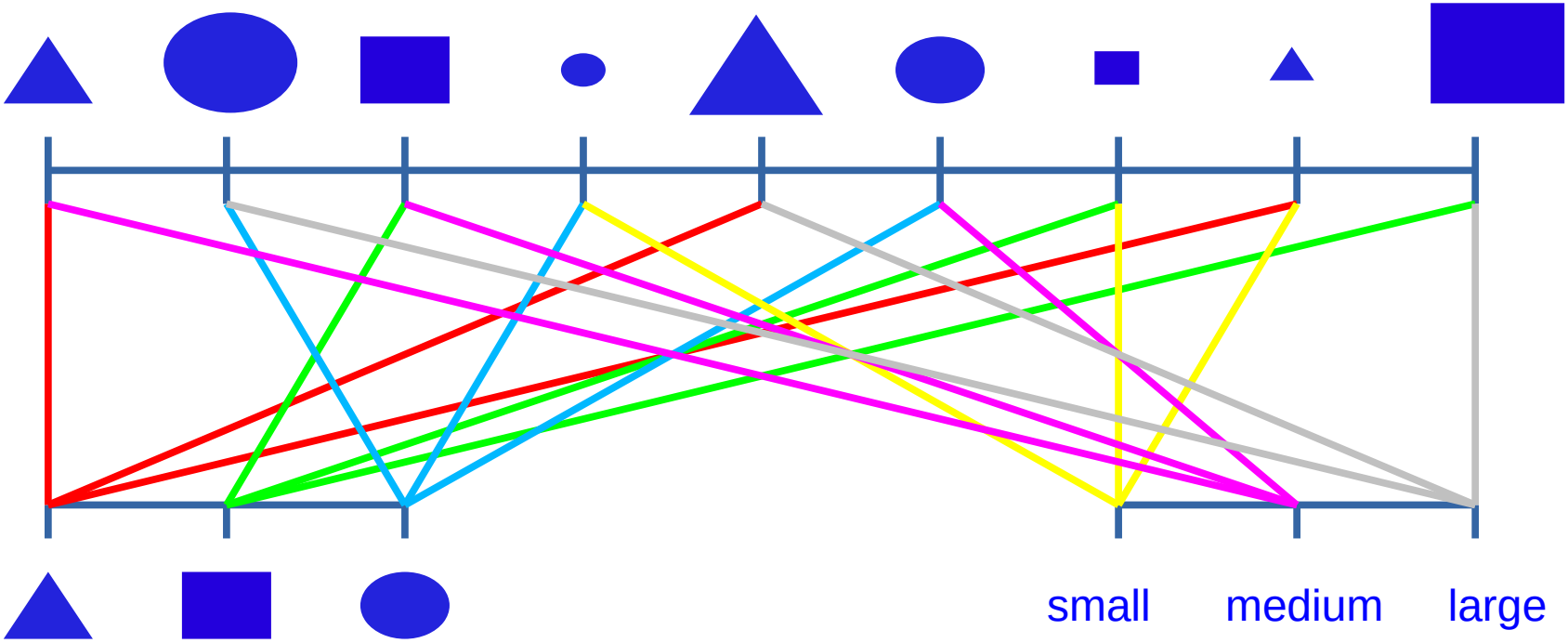
Same Method, withTraining at the Image Level (vs patch)

- **Color indicates orientation (by fitting Gabors)**

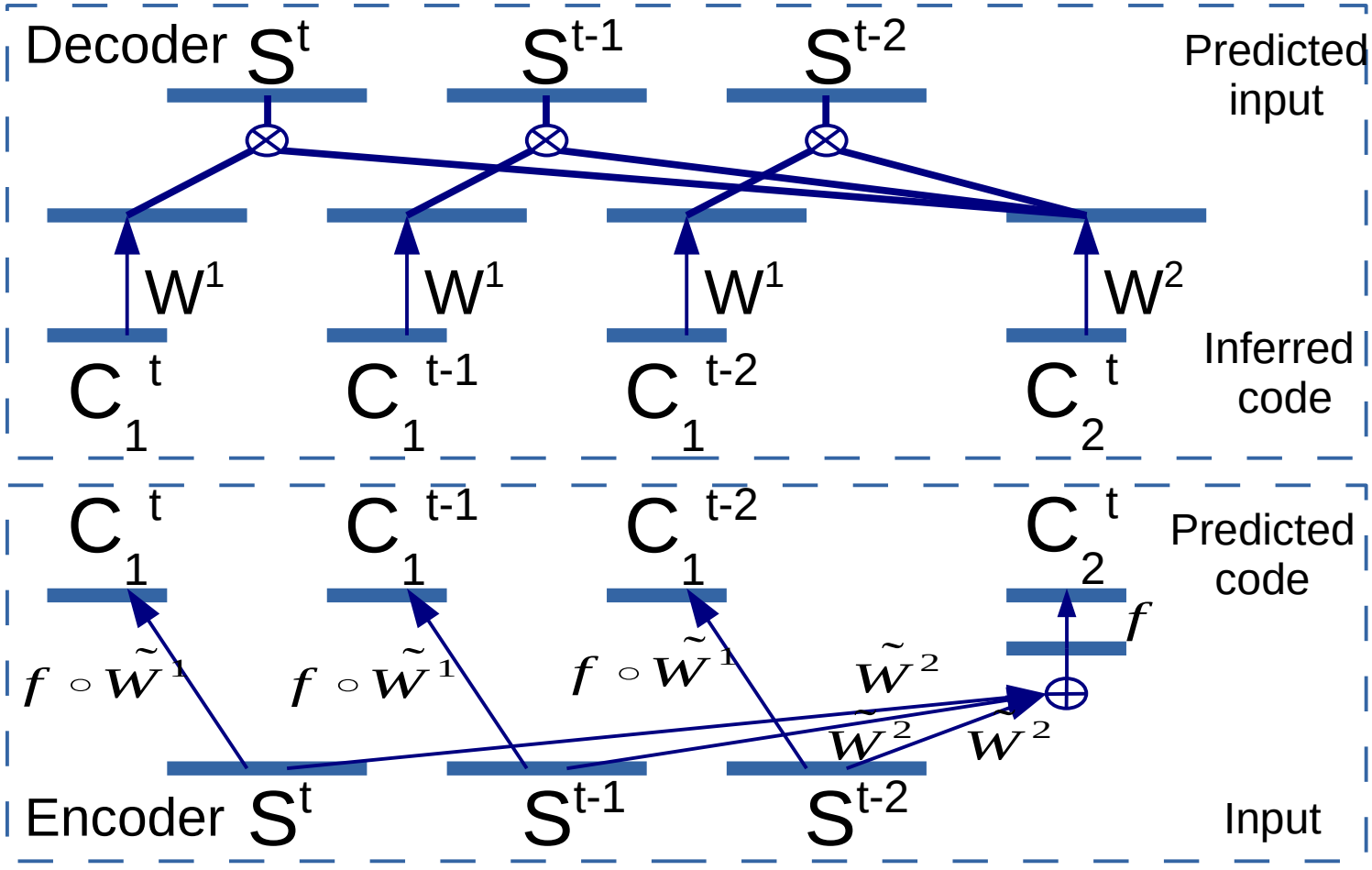


Invariant Features through Temporal Constancy

- **[Gregor & LeCun arXiv:1105.5307]**
  - "Efficient Learning of Sparse Invariant Representations"
- Object is cross-product of object type and instantiation parameters
  - What-where product network: [Hinton 1981] [von der Malsburg 1985]

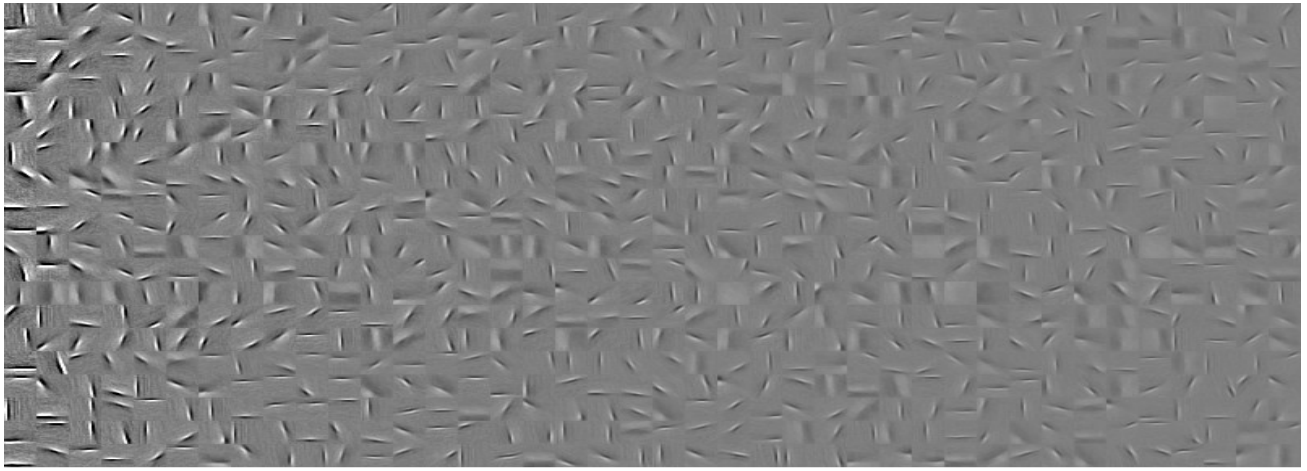


Invariant Features through Temporal Constancy

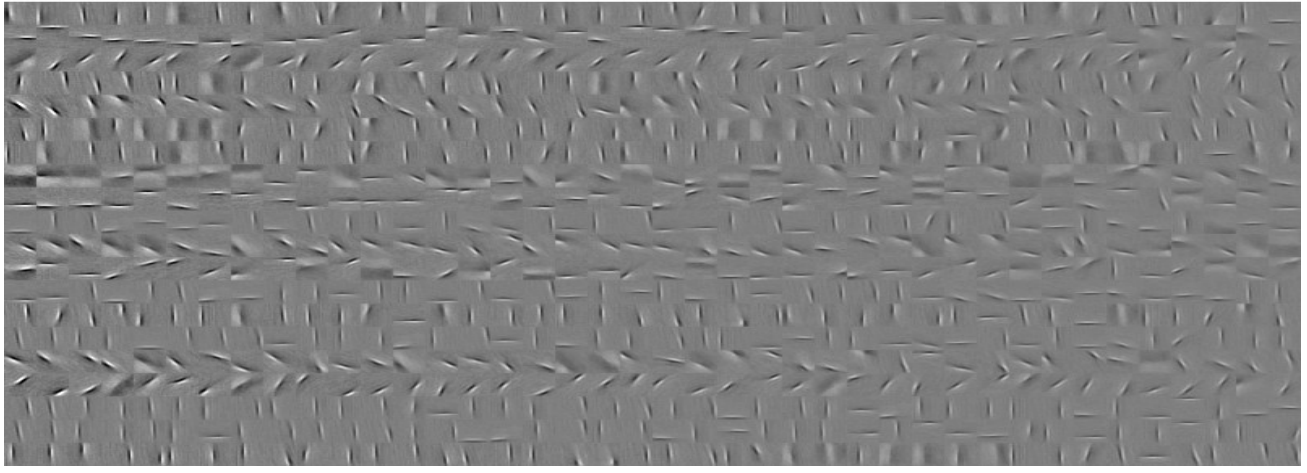


Invariant Features through Temporal Constancy

C1
(where)



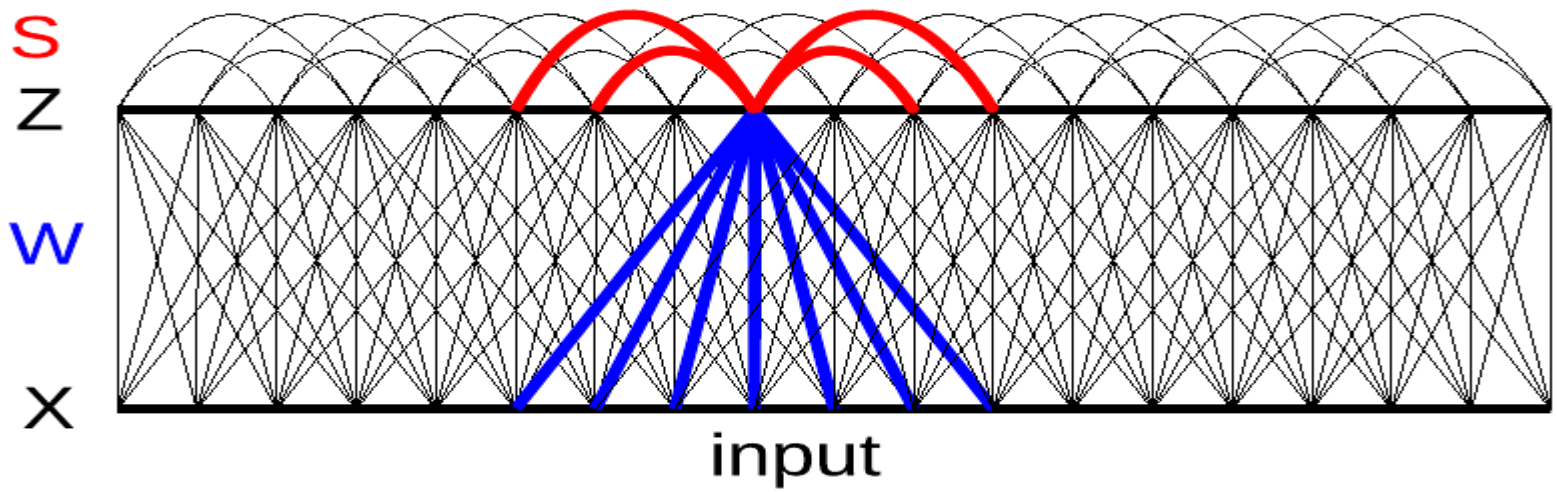
C2
(what)



Invariant Features through Lateral Inhibition

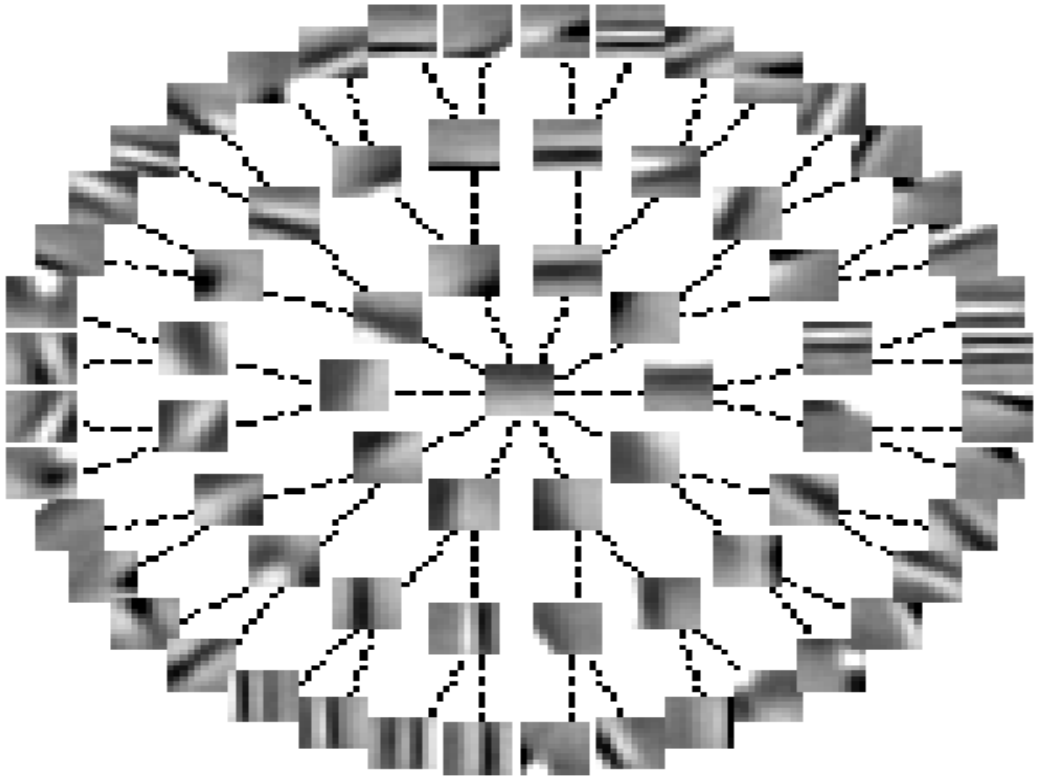
- **[Gregor, Szlam, LeCun NIPS 2011]**
- **Replace the L1 sparsity term by a lateral inhibition matrix**

$$\min_{W,Z} \sum_{x \in X} ||Wz - x||^2 + |z|^T S|z|$$



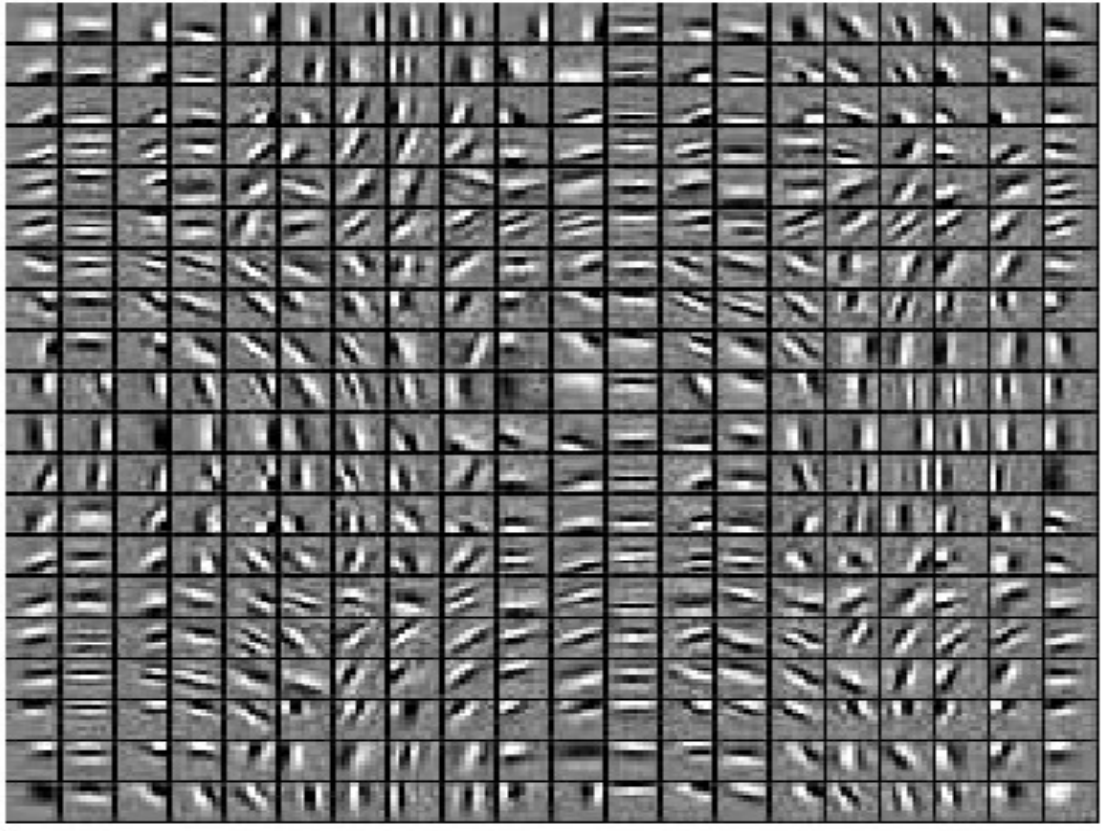
Invariant Features Lateral Inhibition

- **Zeros in S matrix have tree structure**



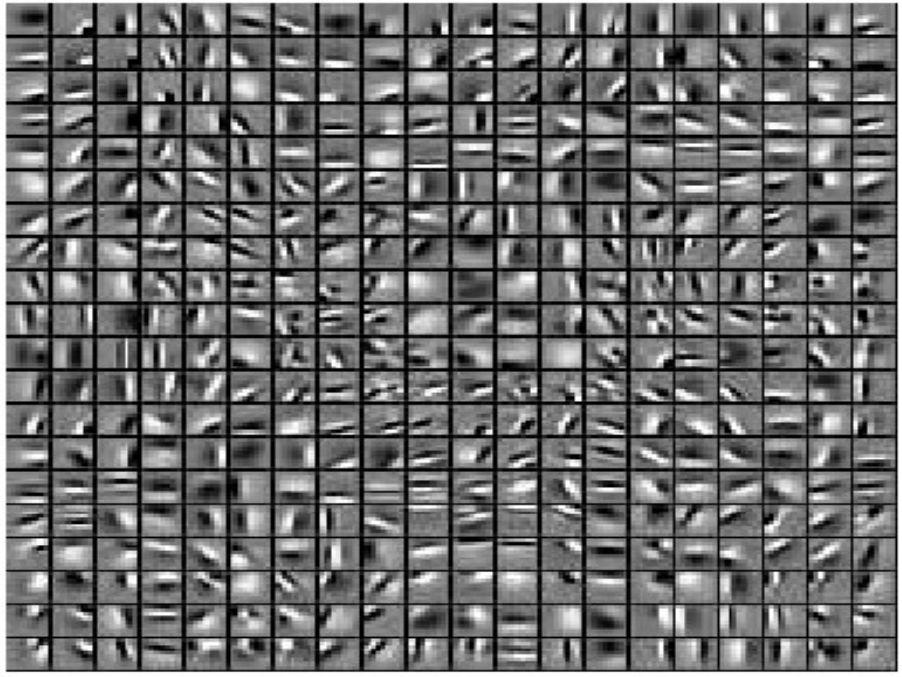
Invariant Features Lateral Inhibition

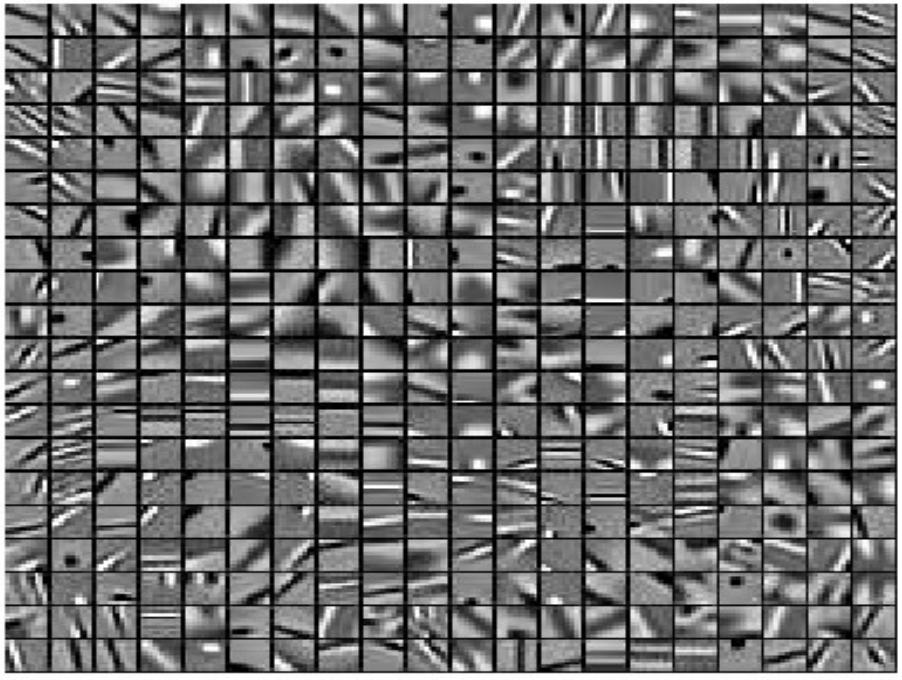
- **Non-zero values in S form a ring in a 2D topology**
  - Input patches are high-pass filtered



Invariant Features Lateral Inhibition

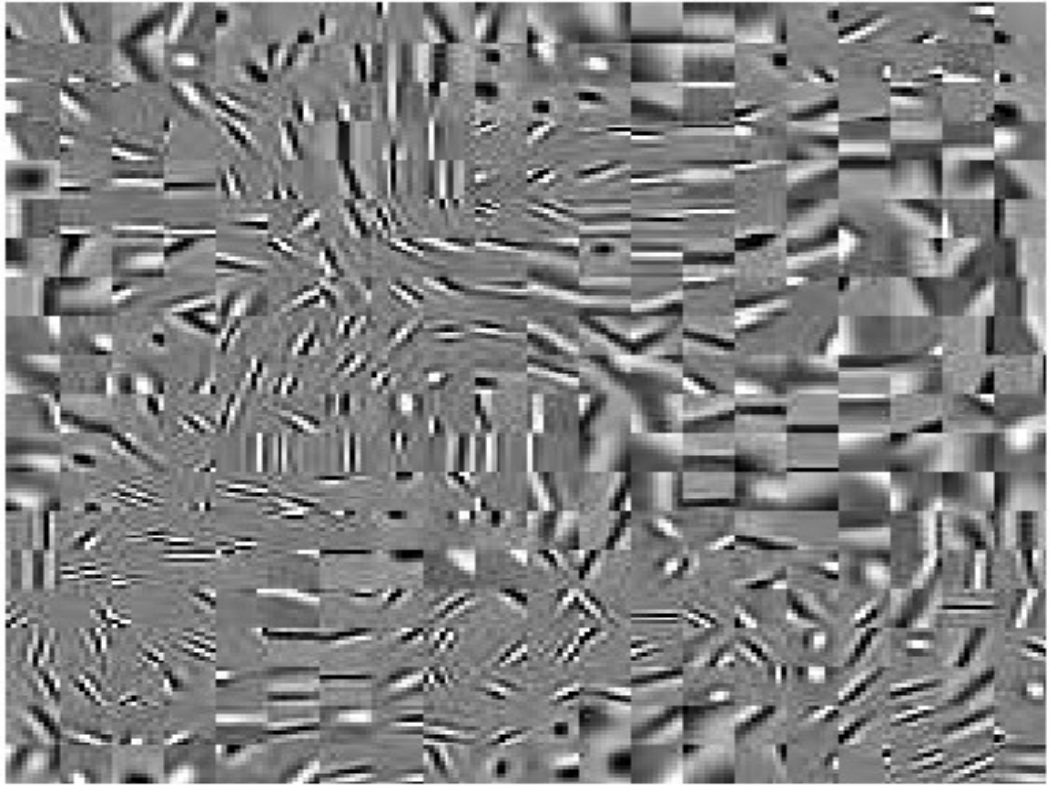
- **Non-zero values in S form a ring in a 2D topology**
  - Left: non high-pass filtering of input
  - Right: patch-level mean removal





Invariant Features Short-Range Lateral Excitation + L1

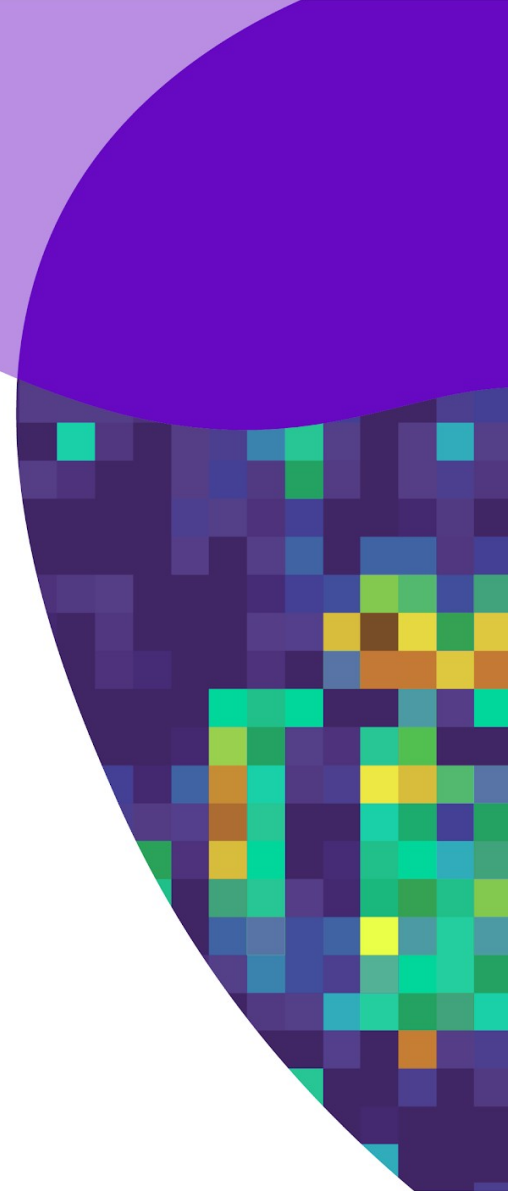
Orientation selectivity with multiple scales



FACEBOOK AI

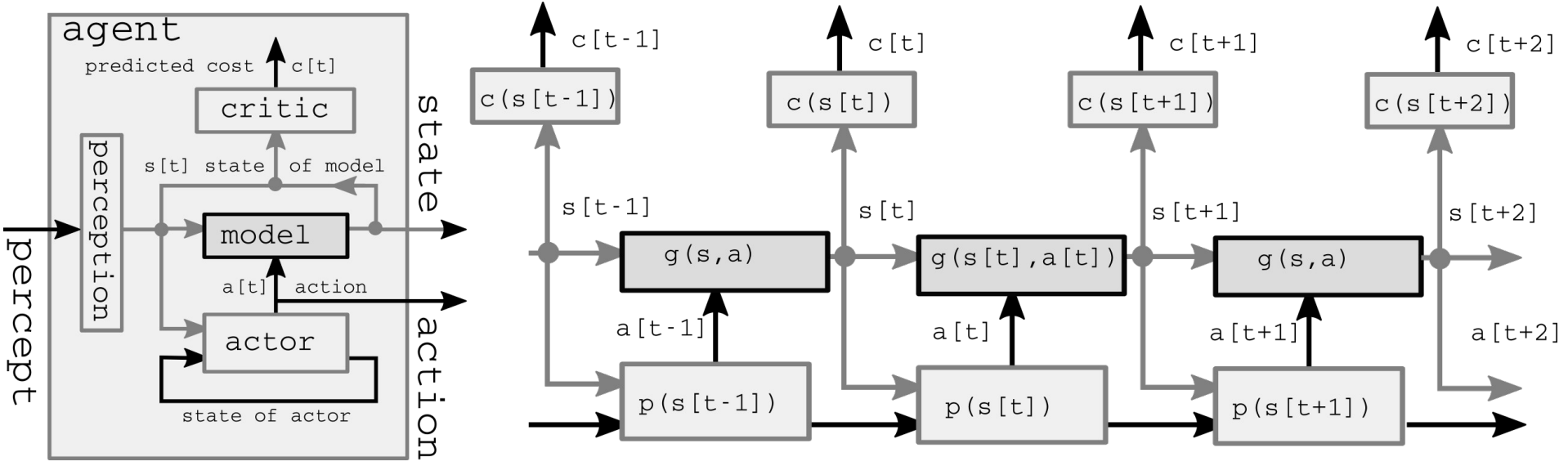
Learning World Models for Autonomous Control

With latent variables to handle the uncertainty in the world.



A Forward Model of the World

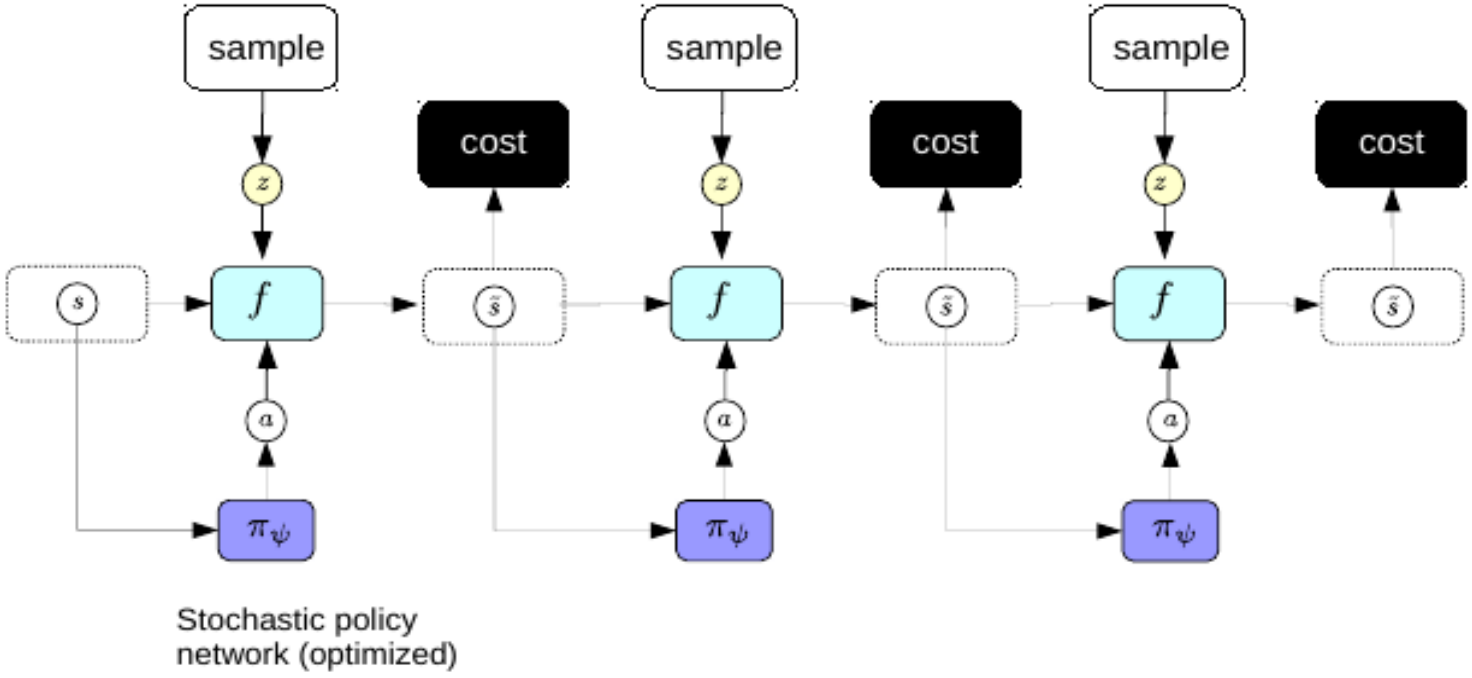
- **Learning forward models for control**
  - s[t+1] = g(s[t], a[t], z[t])
  - Classical optimal control: find a sequence of action that minimize the cost, according to the predictions of the forward model



Planning/learning using a self-supervised predictive world model

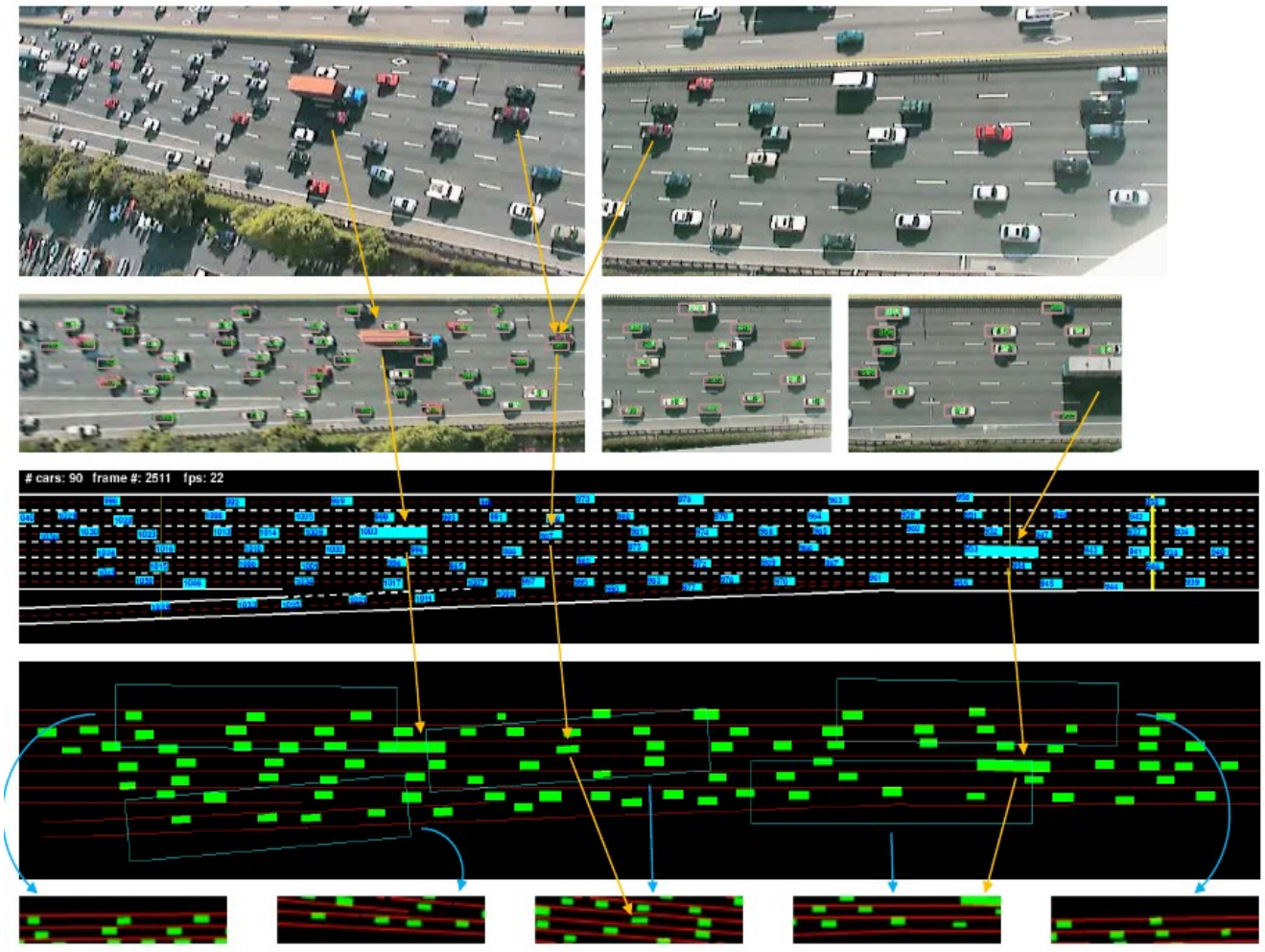
- **Feed initial state**
- **Run the forward model**
- **Backpropagate gradient of cost**
- **Act**
  - (model-predictive control)

  or
- **Use the gradient to train a policy network.**
- **Iterate**



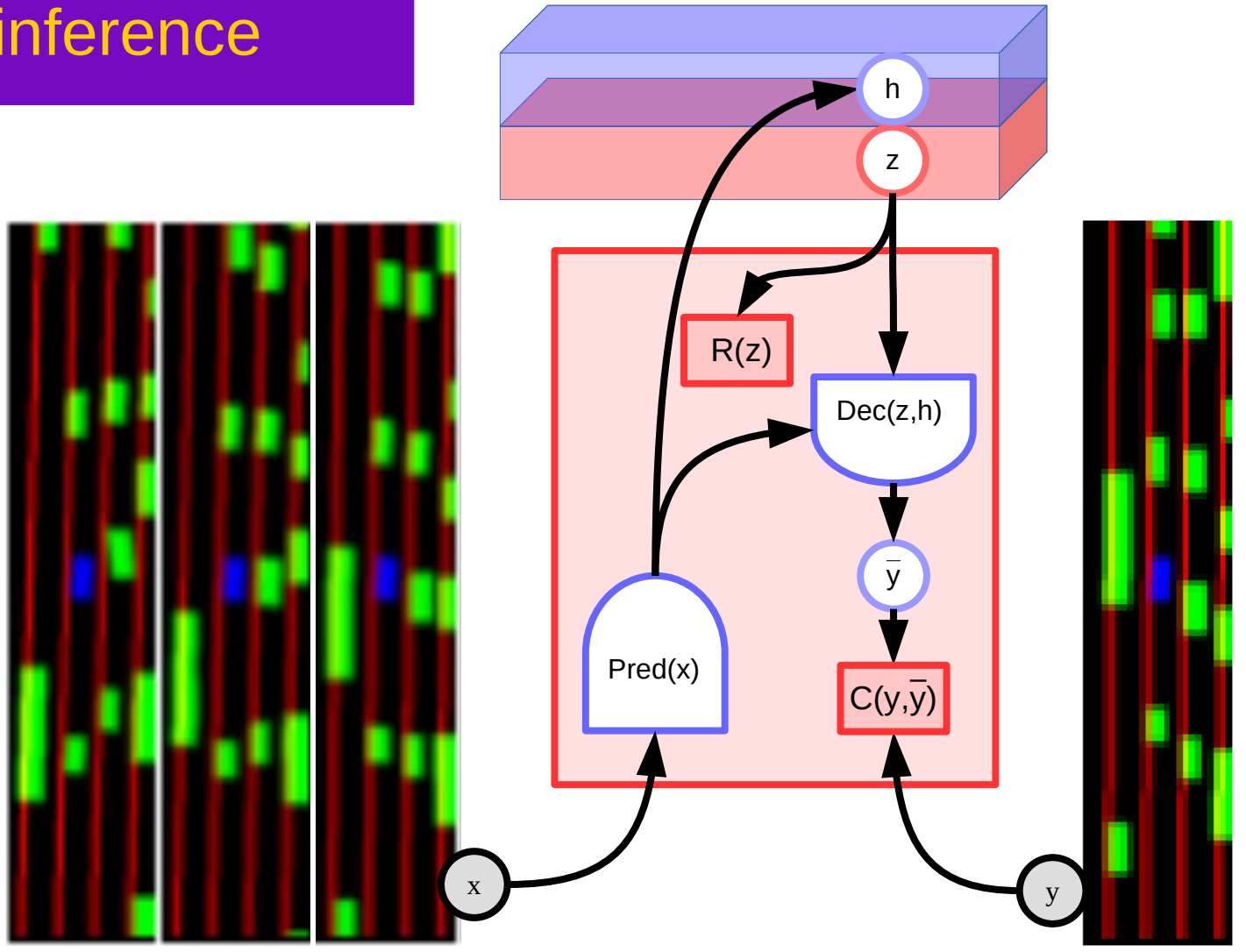
Using Forward Models to Plan (and to learn to drive)

- **Overhead camera on highway.**
  - Vehicles are tracked
- **A "state" is a pixel representation of a rectangular window centered around each car.**
- **Forward model is trained to predict how every car moves relative to the central car.**
  - steering and acceleration are computed



Video Prediction: inference

- **After training:**
  - Observe frames
  - Compute h
  - Sample z
  - Predict next frame

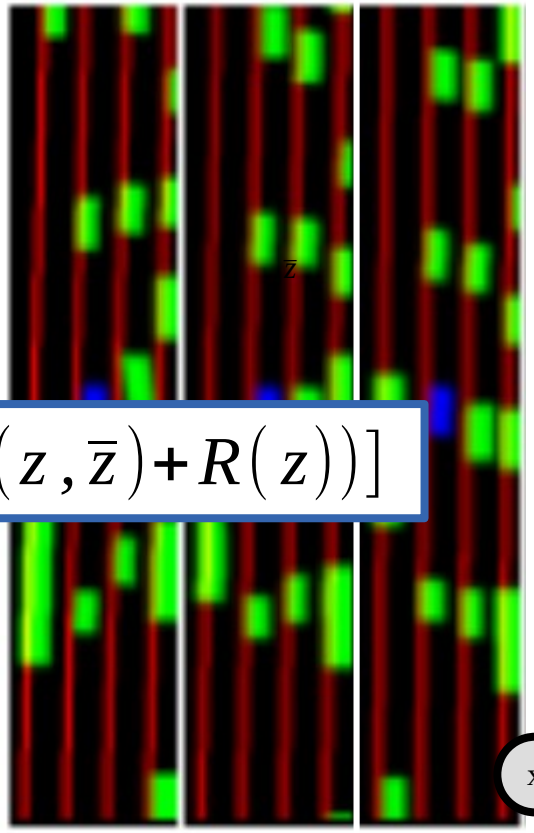


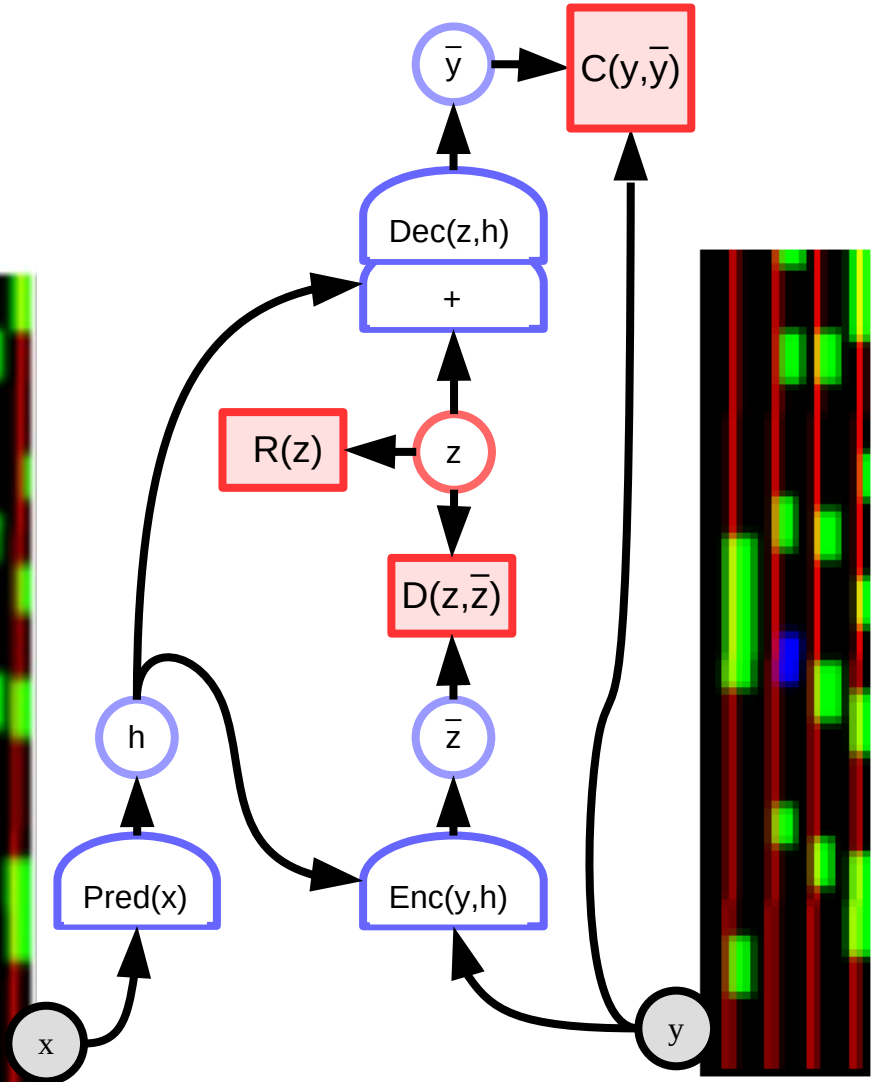
Video Prediction: training

- **Training:**
  - Observe frames
  - Compute h
  - Predict $\bar{z}$ from encoder
  - Sample z, with:

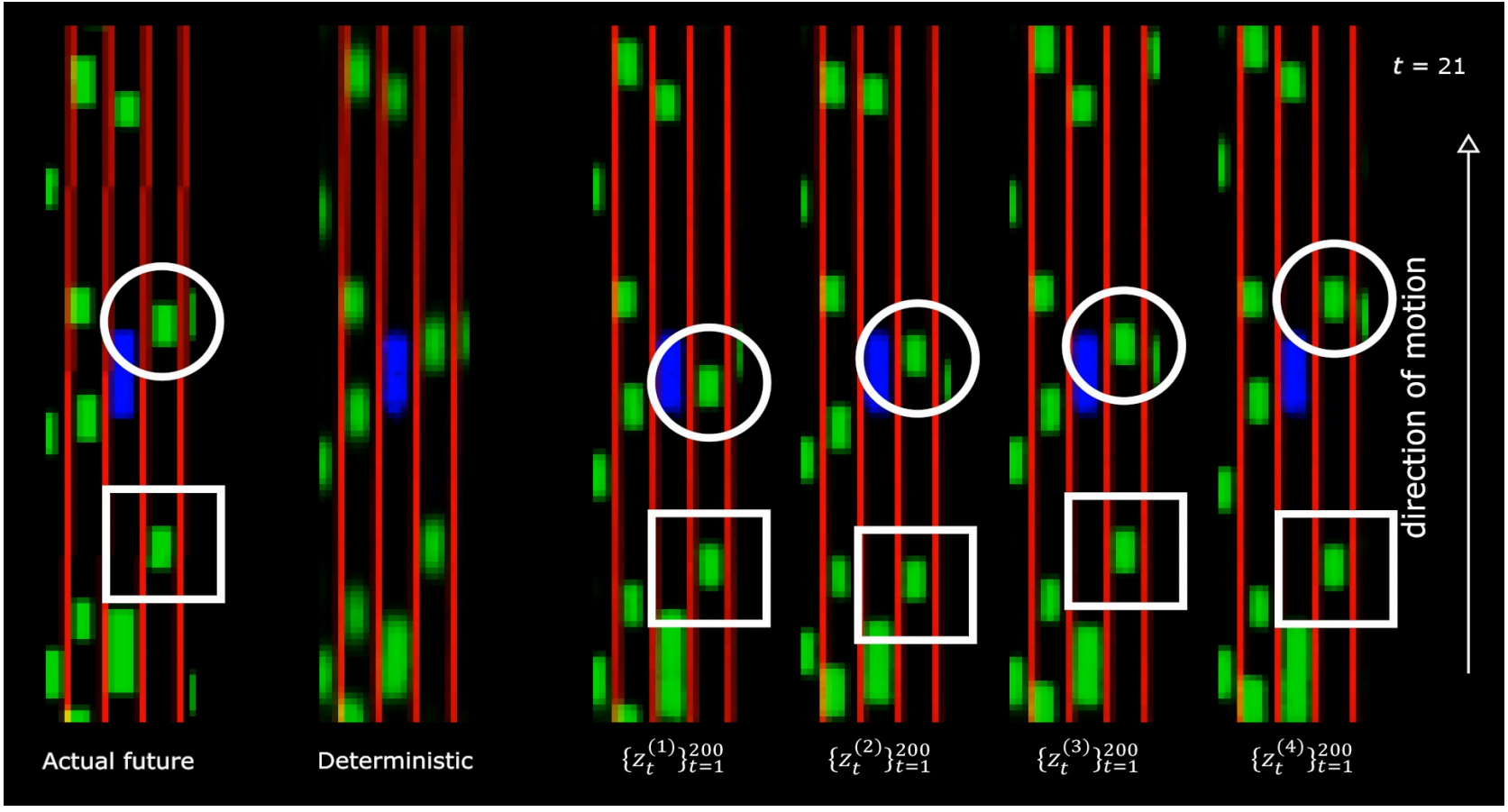
$$P(z/\bar{z}) \propto \exp[-\beta(D(z,\bar{z})+R(z))]$$

  - Predict next frame
  - backprop



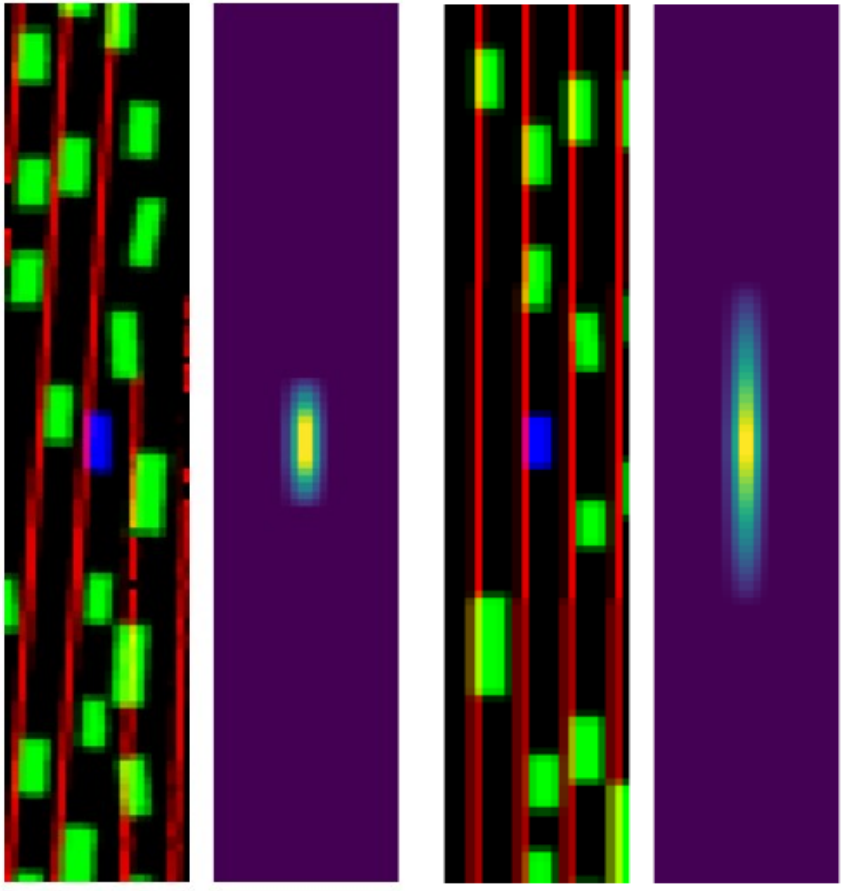


Actual, Deterministic, VAE+Dropout Predictor/encoder



Cost optimized for Planning & Policy Learning

- **Differentiable cost function**
  - Increases as car deviates from lane
  - Increases as car gets too close to other cars nearby in a speed-dependent way
- **Uncertainty cost:**
  - Increases when the costs from multiple predictions (obtained through sampling of drop-out) have high variance.
  - Prevents the system from exploring unknown/unpredictable configurations that may have low cost.

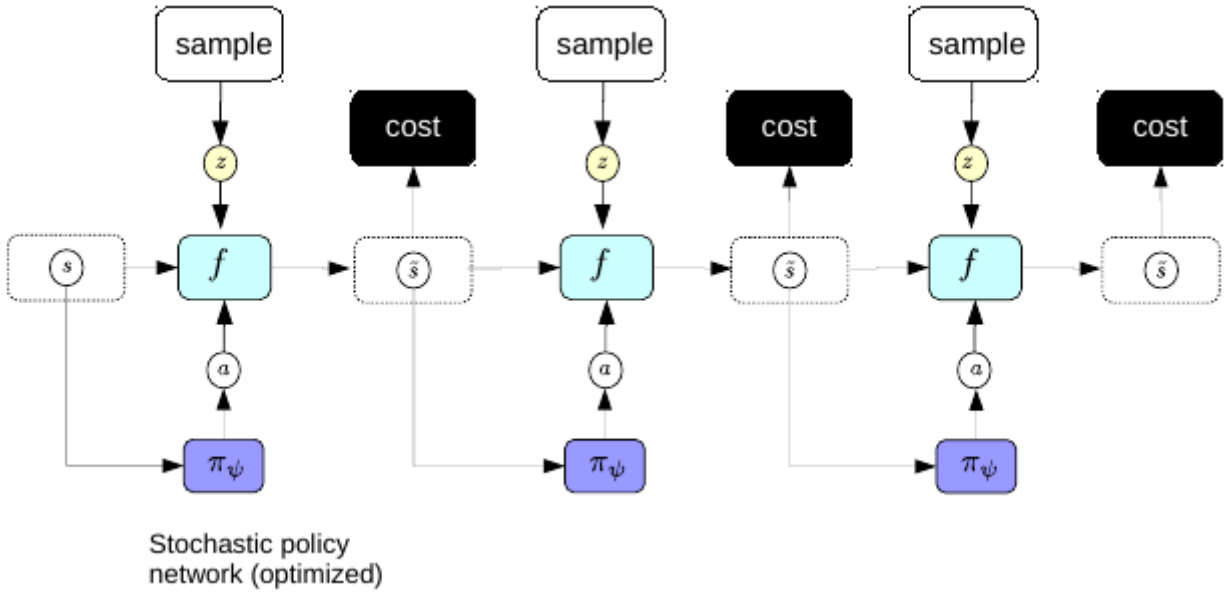


(a) 19.8 km/h (b) 50.3 km/h

Learning to Drive by Simulating it in your Head

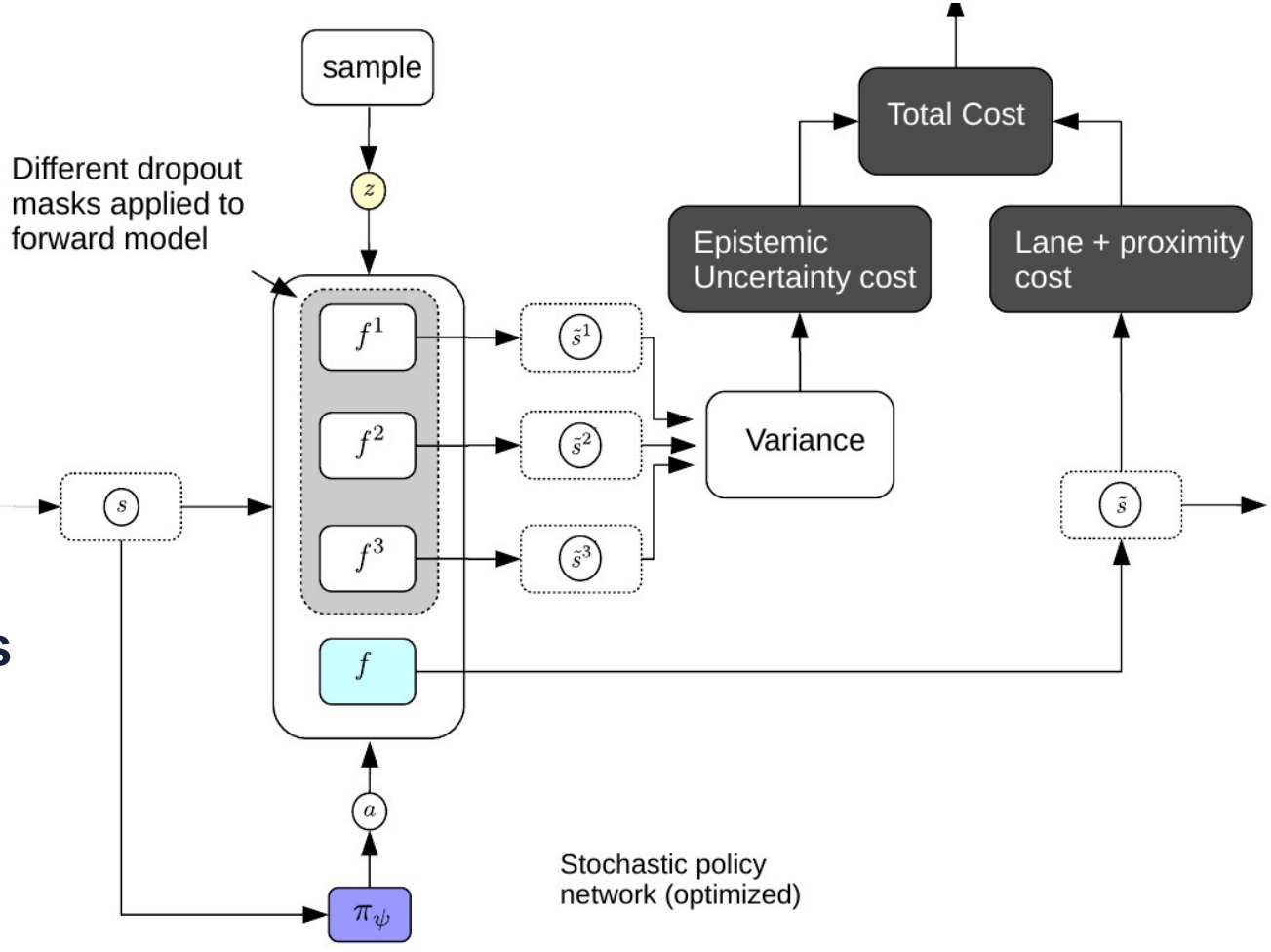
- Feed initial state
- Sample latent variable sequences of length 20
- Run the forward model with these sequences
- Backpropagate gradient of cost to train a policy network.
- Iterate

- No need for planning at run time.

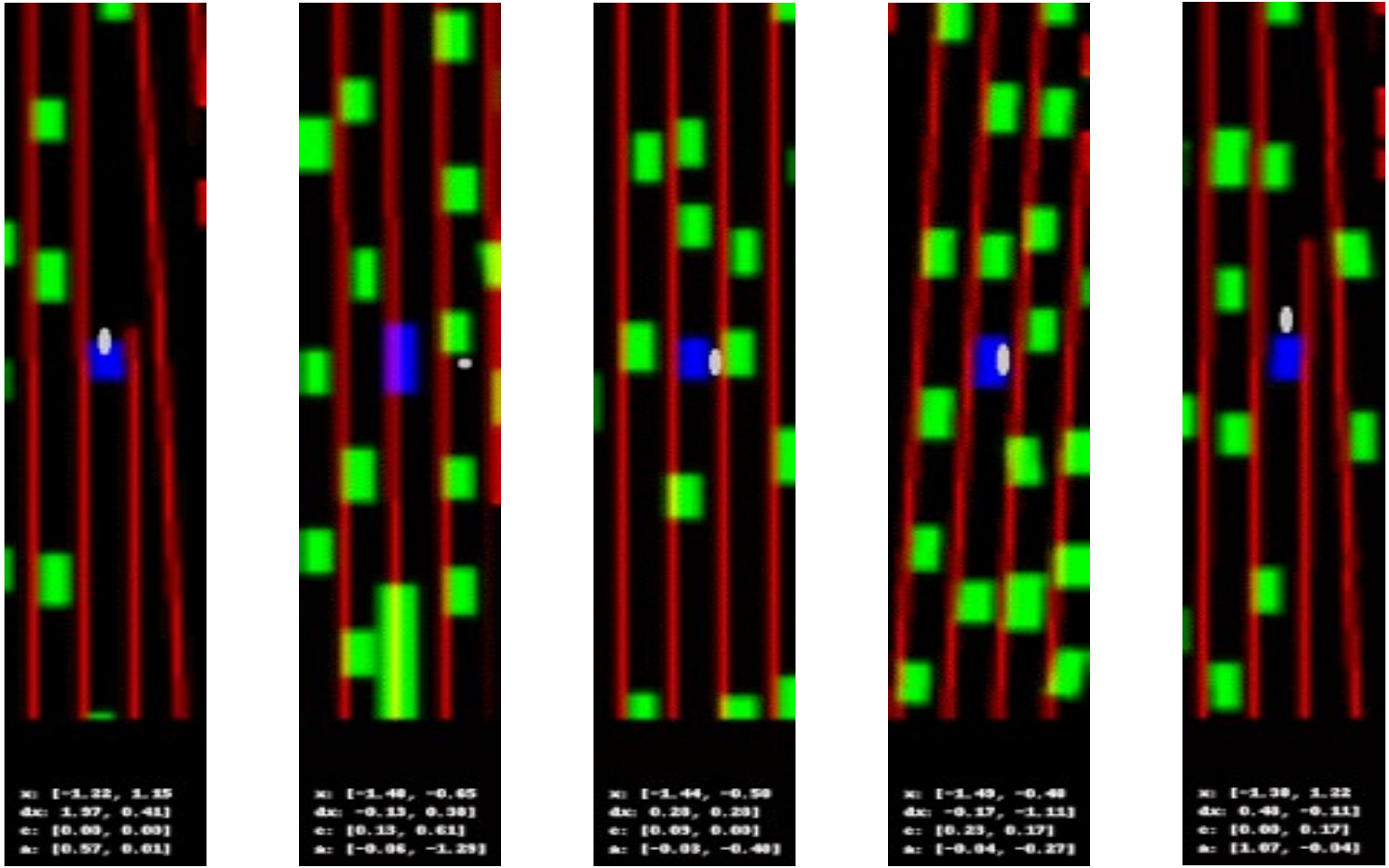


Adding an Uncertainty Cost (doesn't work without it)

- **Estimates epistemic uncertainty**
- **Samples multiple drop-puts in forward model**
- **Computes variance of predictions (differentiably)**
- **Train the policy network to minimize the lane&proximity cost plus the uncertainty cost.**
- **Avoids unpredictable outcomes**

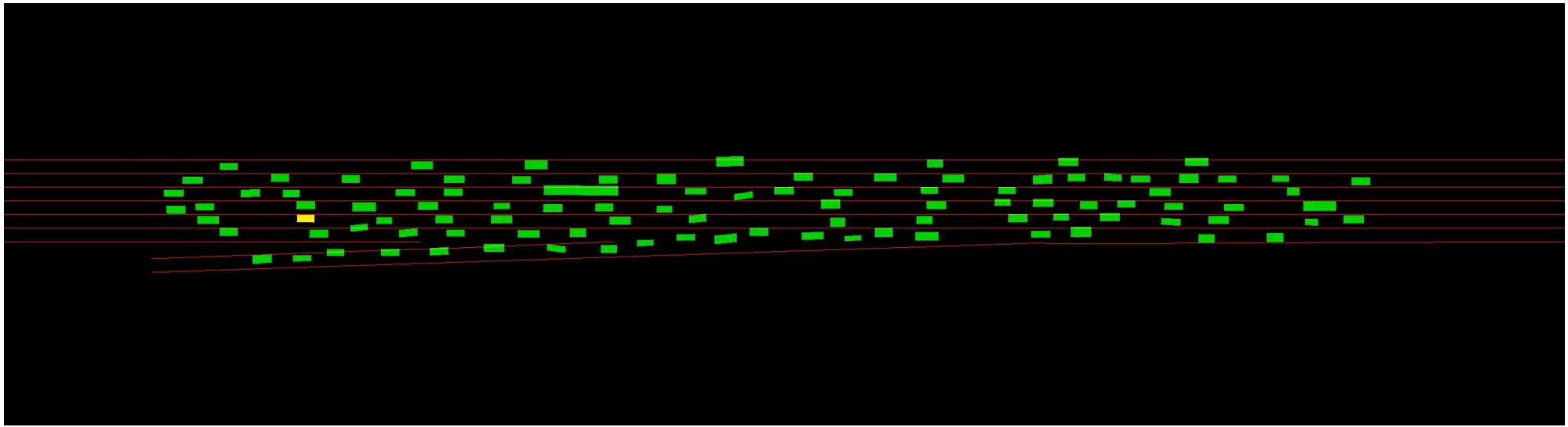


Driving an Invisible Car in “Real” Traffic



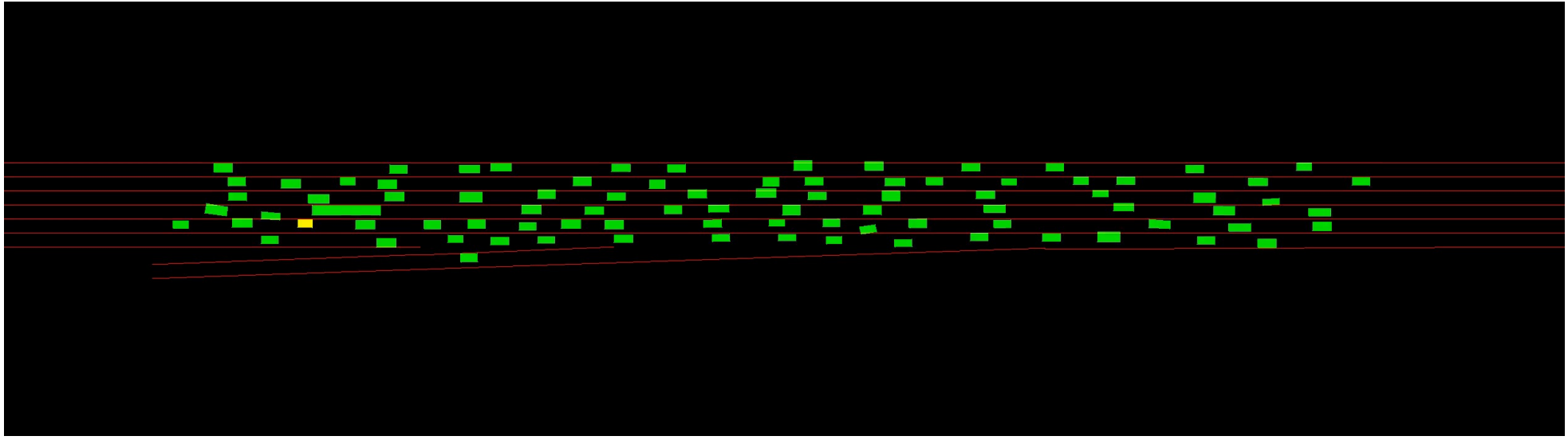
Driving!

- **Yellow: real car**
- **Blue: bot-driven car**



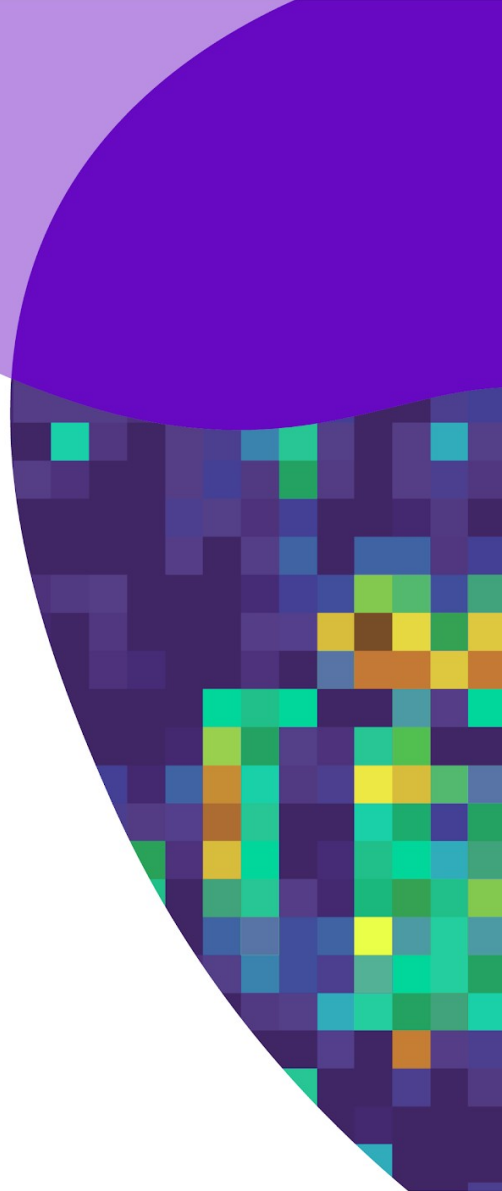
Driving!

- Yellow: real car
- Blue: bot-driven car



FACEBOOK AI

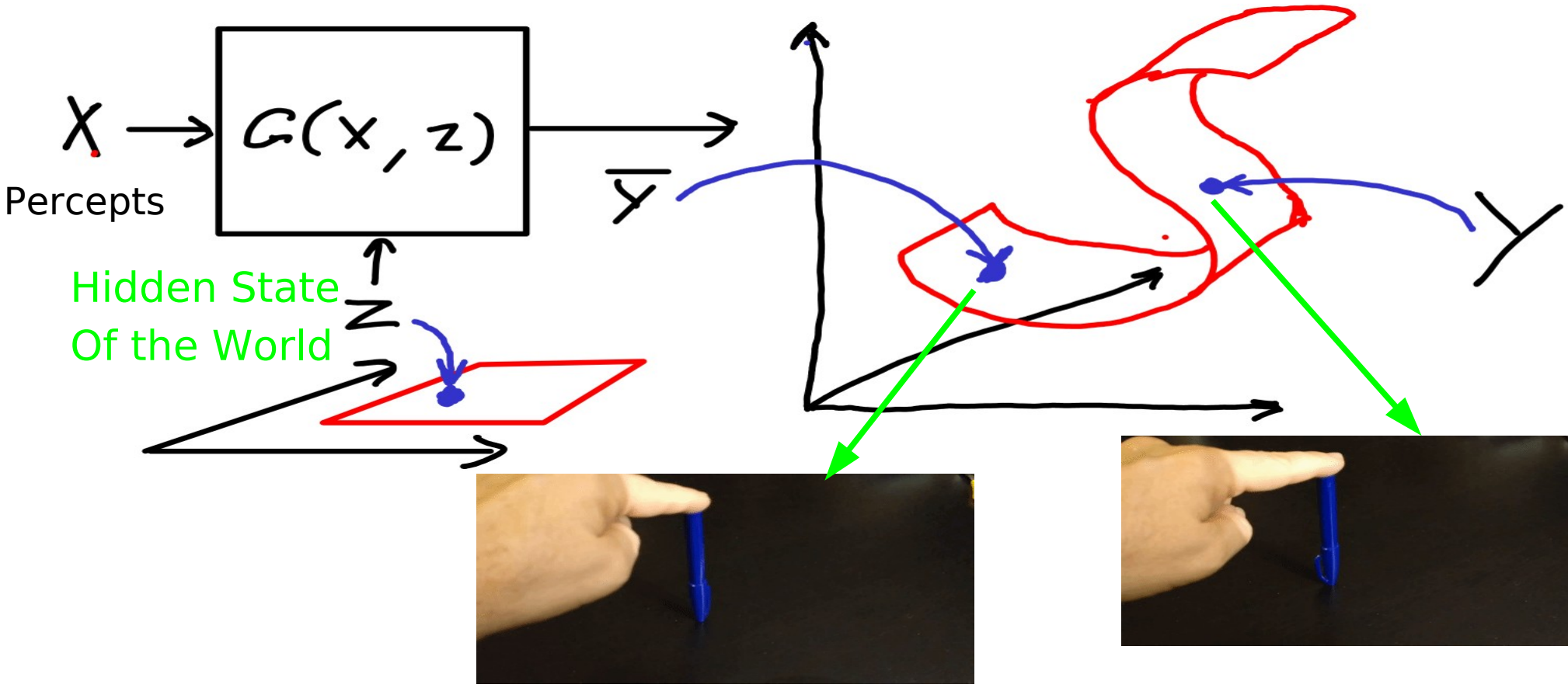
Generative Adversarial Networks



Contrastive method in which a generator network produces the contrastive samples

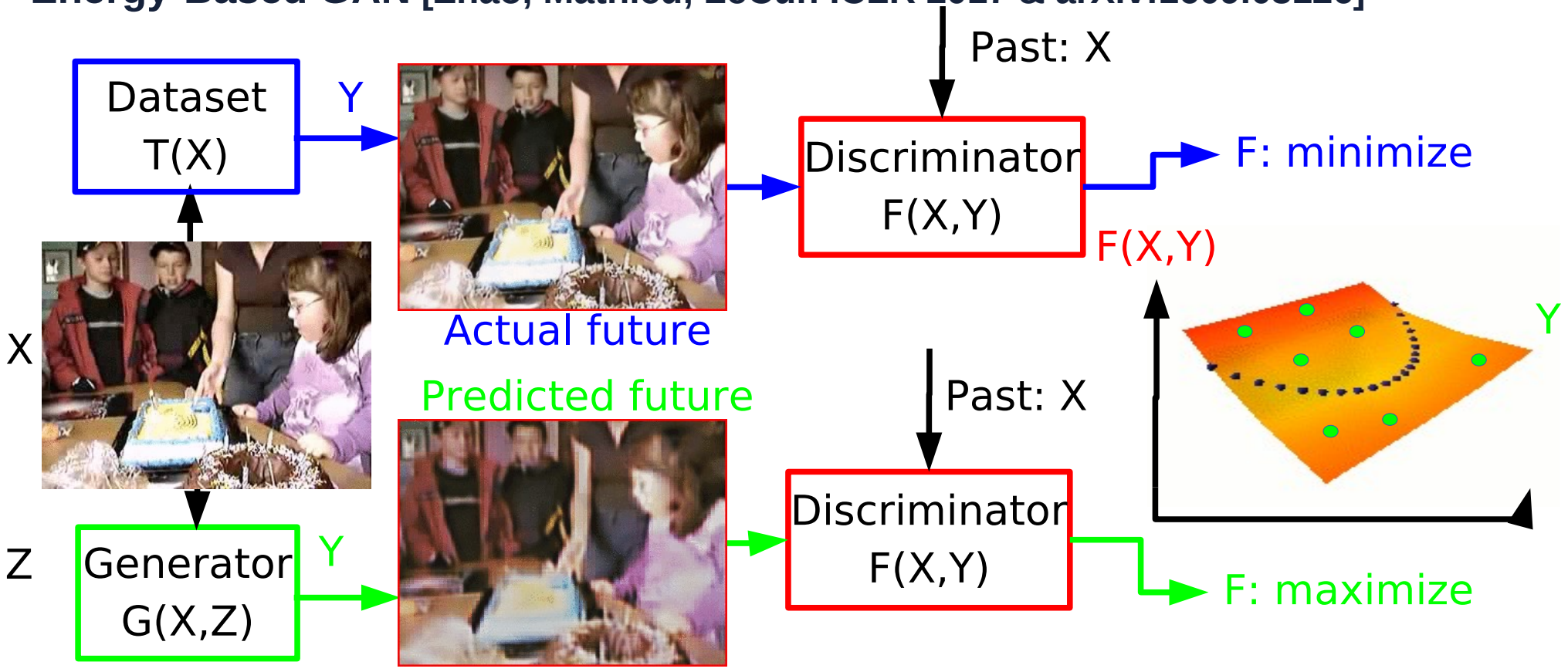
The Hard Part: Prediction Under Uncertainty

- **Invariant prediction: The training samples are merely representatives of a whole set of possible outputs (e.g. a manifold of outputs).**



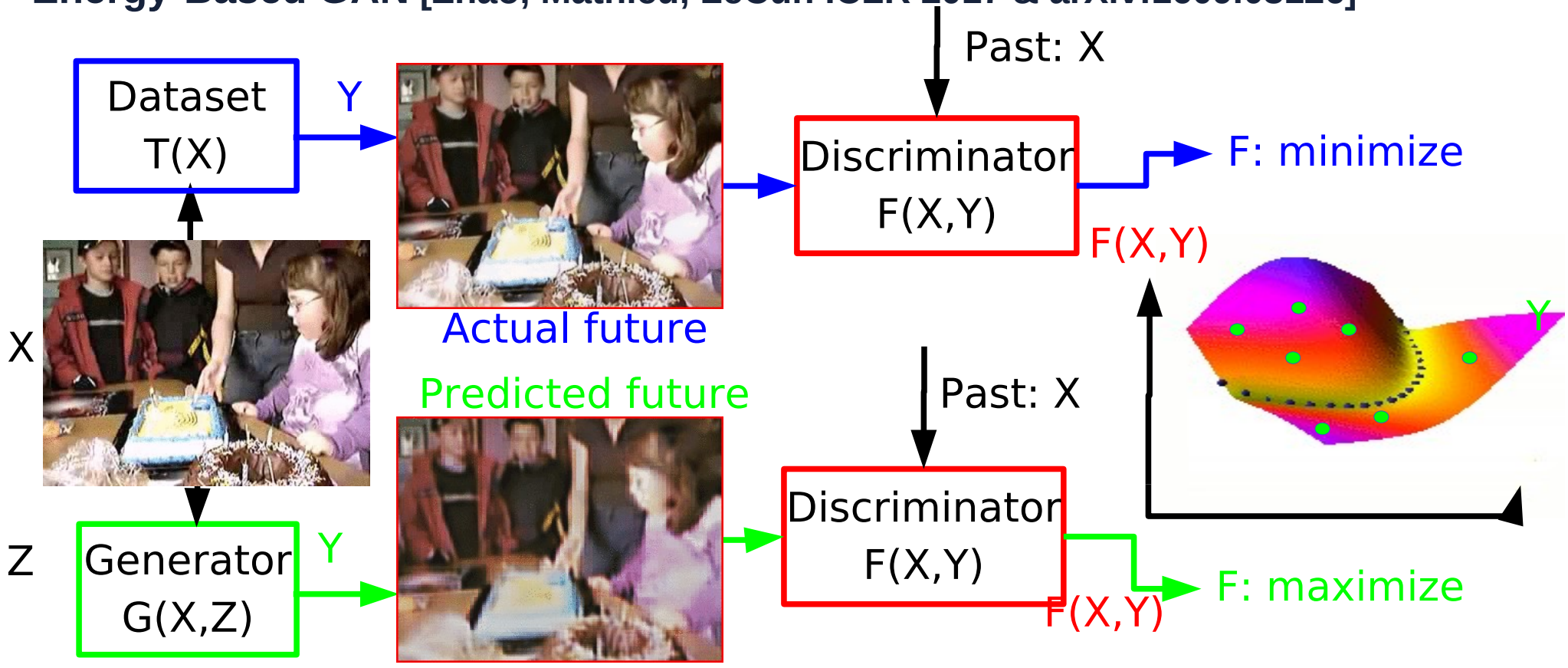
Adversarial Training: the key to prediction under uncertainty?

- **Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],**
- **Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]**



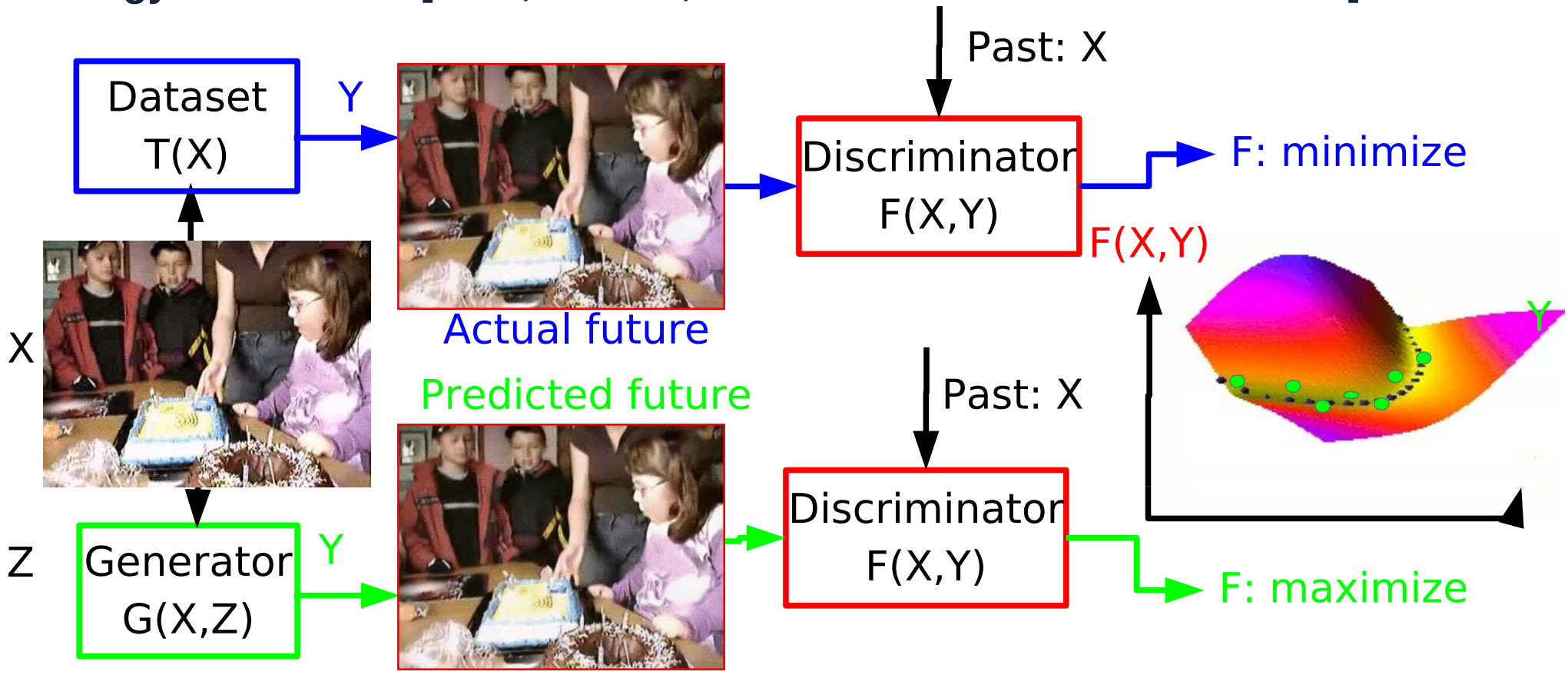
Adversarial Training: the key to prediction under uncertainty?

- **Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],**
- **Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]**



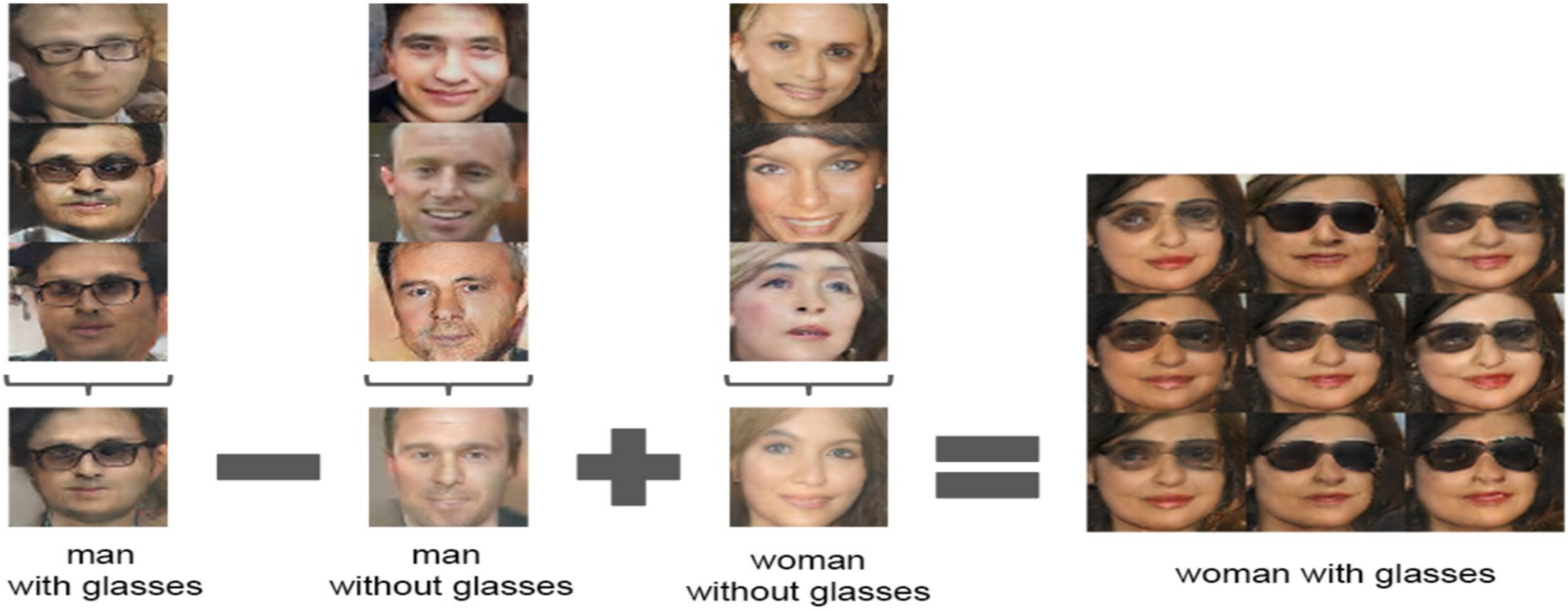
Adversarial Training: the key to prediction under uncertainty?

- **Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],**
- **Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]**



Face Algebra (in DCGAN space)

- **DCGAN: adversarial training to generate images.**
  - [Radford, Metz, Chintala 2015]



Faces “invented” by a GAN (Generative Adversarial Network)

- **Random vector → Generator Network → output image** [Goodfellow NIPS 2014]
  [Karras et al. ICLR 2018] (from NVIDIA)

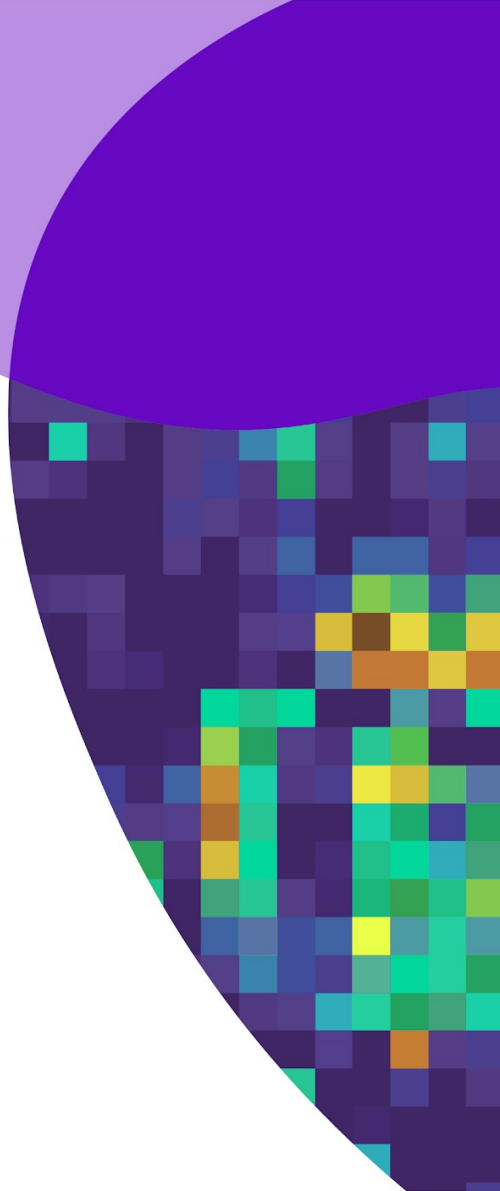
Self-supervised Adversarial Learning for Video Prediction

- **Our brains are “prediction machines”**
- **Can we train machines to predict the future?**
- **Some success with “adversarial training”**
  - [Mathieu, Couprie, LeCun arXiv:1511:05440]
- **But we are far from a complete solution.**

FACEBOOK AI

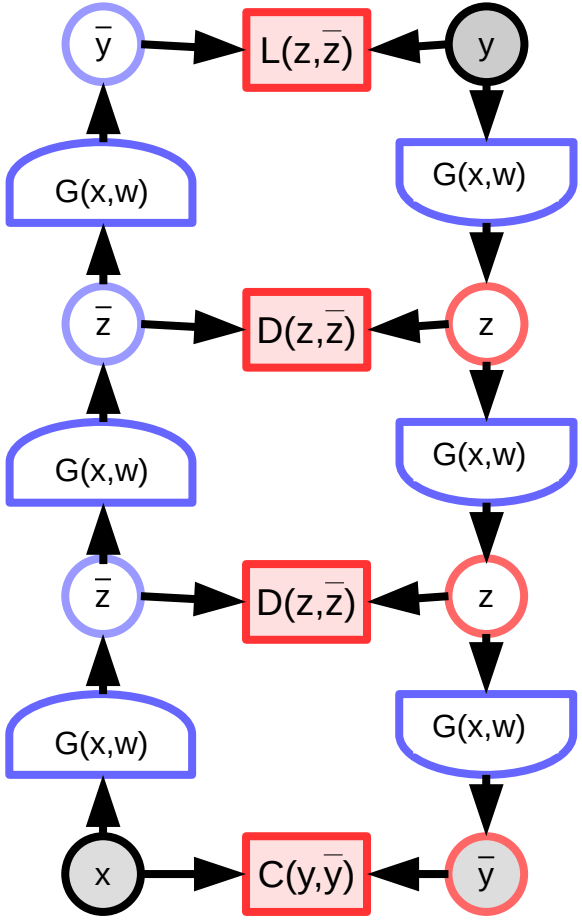
Stacked (regularized) Auto-Encoders

Training feature hierarchies



Stacking AE to learn hierarchical features?

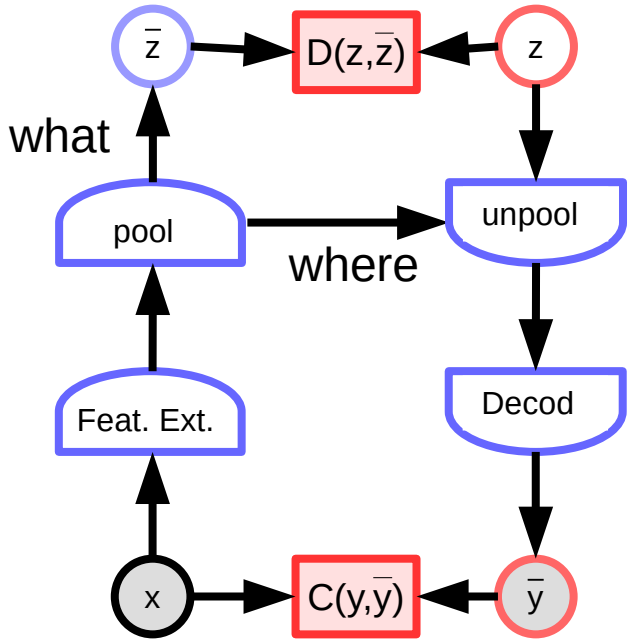
- **Classifiers are many-to-one (non injective) functions.**
  - Because they have invariances
- **They are not invertible**
  - Every stage of a classifier eliminates information about the input
  - And preserves information for the task.



What-Where AE

- **Whenever a non-reversible function is applied, keep the complementary information to allow reconstruction**

- **Example: pooling**
  - $h^* = \max_i(h_i)$: “what”
  - $i^* = \text{argmax}_i(hi)$: “where”
- **[Ranzato 2007], [Gregor 2011], Hinton's Capsules**
- **Very old ideas by Hinton & Zemel, von der Malsburg and others about separating identity from instantiation parameters.**

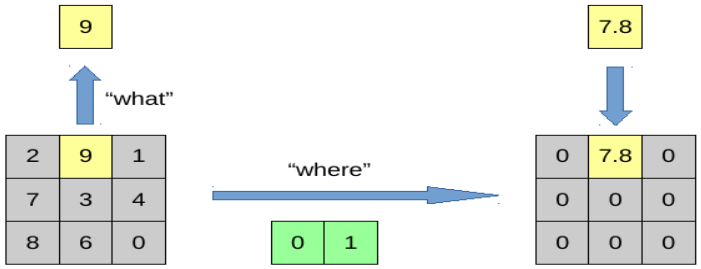


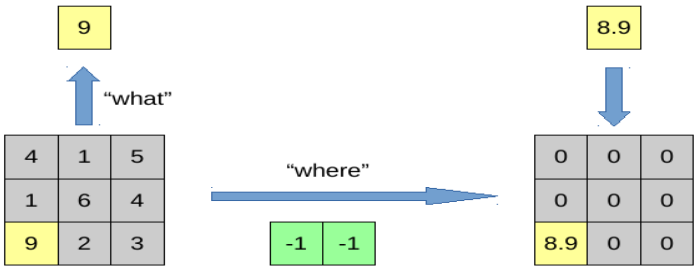
Computing the "where": Phase Pooling

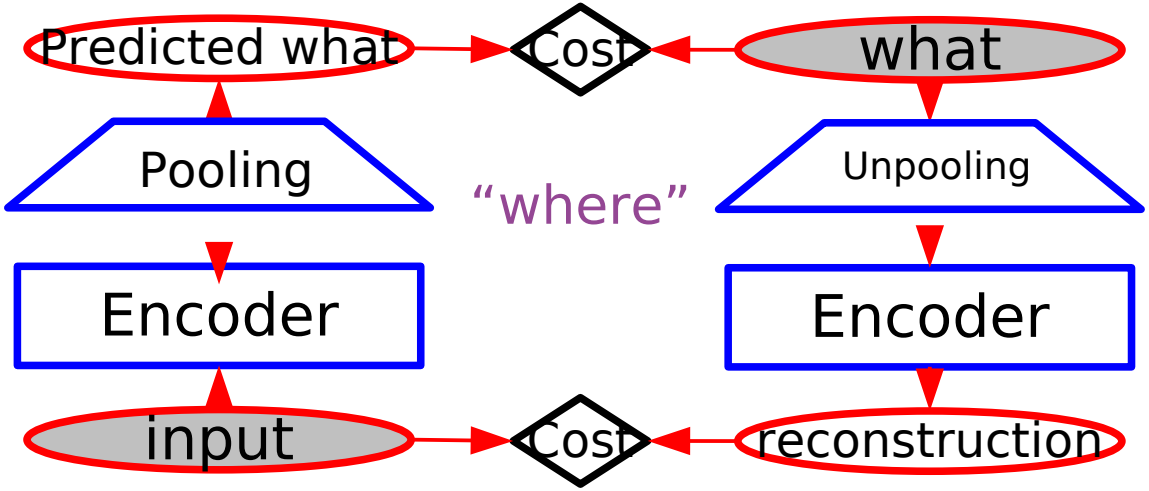
A funny kind of pooling/unpooling

$$m_k = \sum_{N_k} z(f,x,y) \frac{e^{\beta z(f,x,y)}}{\sum_{N_k} e^{\beta z(f',x',y')}} \approx \max_{N_k} z(f,x,y)$$

$$\mathbf{p}_k = \sum_{N_k} \begin{bmatrix} f \\ x \\ y \end{bmatrix} \frac{e^{\beta z(f,x,y)}}{\sum_{N_k} e^{\beta z(f',x',y')}} \approx \arg\max_{N_k} z(f,x,y)$$

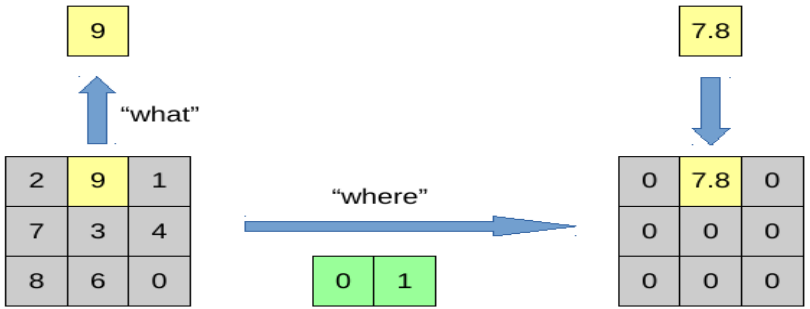




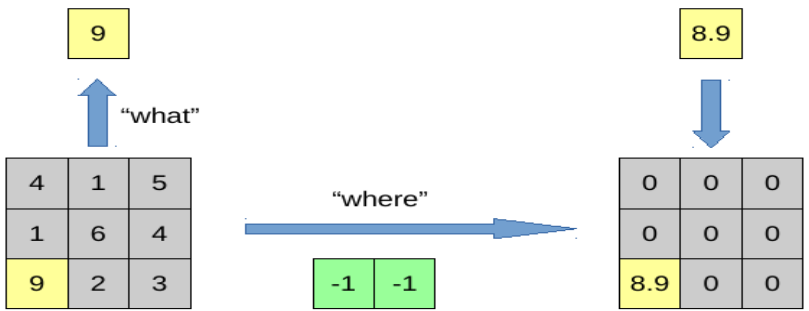


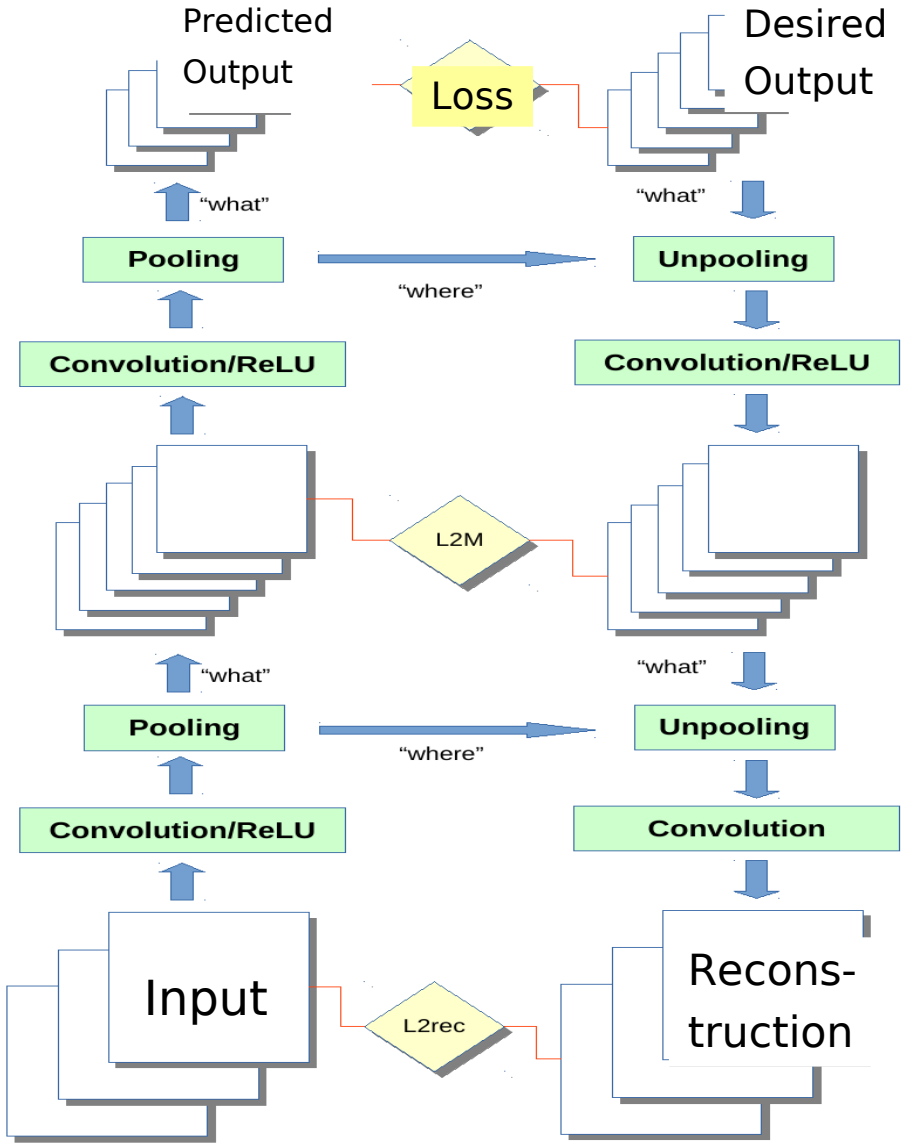
Stacked What-Where Auto-Encoder (SWWAE)

- [Zhao, Mathieu, LeCun arXiv:1506.0235
- **Stacked What-Where Auto-Encoder**



- **A bit like a ConvNet paired with a DeConvNet**





SWWAE: Reconstructions with Unpooling

- **Network: MNNIST → [Conv 5x5] → 16 fmaps → Conv 3x3 → 32 fmaps → Pooling PxP**
- **Trained unsupervised. Hard max-pooling at test time.**

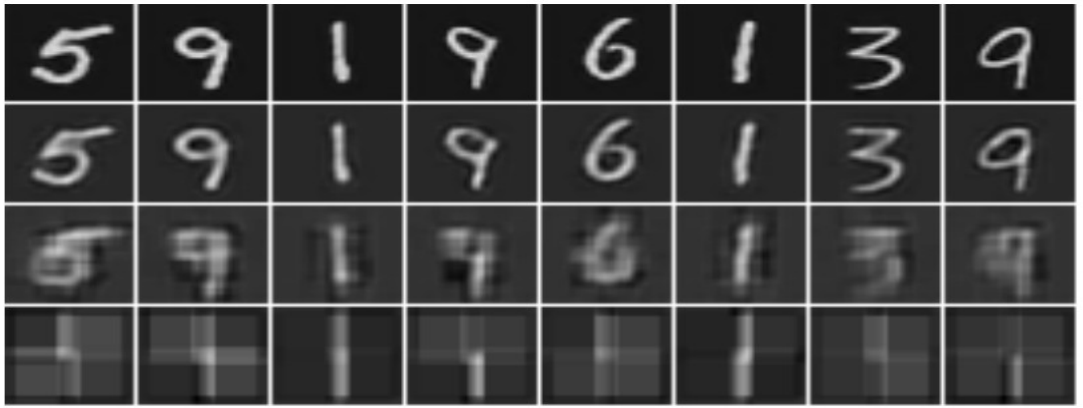
Upsampling

P=2

P=4

P=8

P=16



Unpooling

P=2

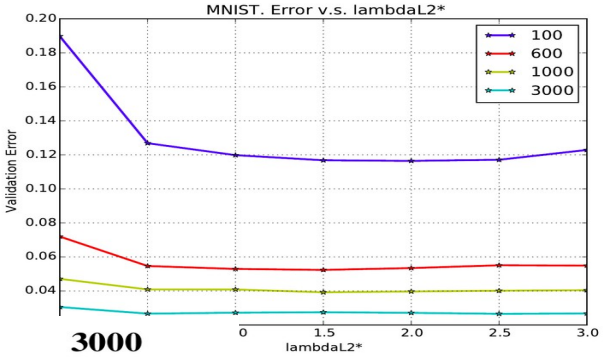
P=4

P=8

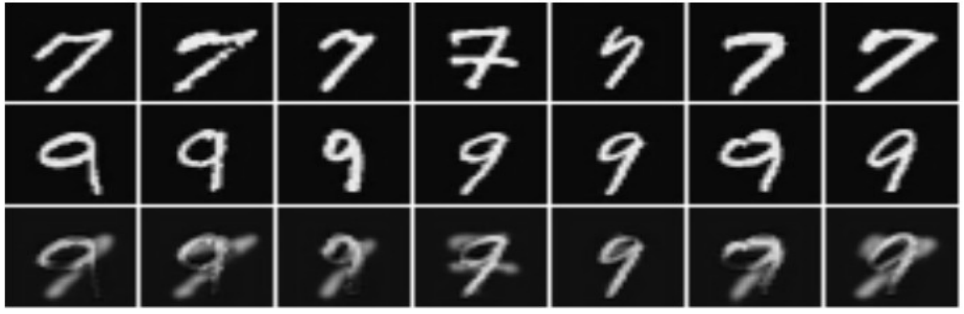
P=16

MNIST: Recognition & Generation

model / N	100	600	1000	3000
SWWAE	[1]11.65 ± 0.20	[1]5.24 ± 0.06	[1]3.92 ± 0.09	[2]2.66 ± 0.07
dp	21.11 ± 0.65	7.11 ± 0.27	5.34 ± 0.39	3.23 ± 0.30
dp-fc	16.09 ± 0.96	6.14 ± 0.36	4.368 ± 0.50	2.98 ± 0.08
unsup-sfx	17.81 ± 0.06	8.41 ± 0.08	6.40 ± 0.06	4.76 ± 0.03
unsup-pretr	-	9.80 ± 0.06	6.135 ± 0.03	4.41 ± 3.11
noL2M	[2]13.48 ± 2.35	[2]5.69 ± 0.33	[2]3.97 ± 0.37	[1]2.52 ± 0.06



model / N	100	600	1000	3000
Convnet (LeCun et al. (1998))	22.98	7.86	6.45	3.35
TSVM (Vapnik & Vapnik (1998))	16.81	6.16	5.38	3.45
CAE (Rifai et al. (2011b))	13.47	6.3	4.77	3.22
MTC (Rifai et al. (2011a))	12.03	5.13	3.64	2.57
PL-DAE (Lee (2013))	10.49	5.03	3.46	2.69
WTA-AE (Makhzani & Frey (2014))	-	**2.37**	1.92	-
M1+M2 (Kingma et al. (2014))	3.33 ± 0.14	2.59 ± 0.05	2.40 ± 0.02	2.18 ± 0.04
LadderNetwork (Rasmus et al. (2015a))	**1.13 ± 0.04**	-	**1.00 ± 0.06**	-
SWWAE without dropout	9.17 ± 0.11	4.16 ± 0.11	3.39 ± 0.01	2.50 ± 0.01
SWWAE with dropout	8.71 ± 0.34	3.31 ± 0.40	2.83 ± 0.10	**2.10 ± 0.22**

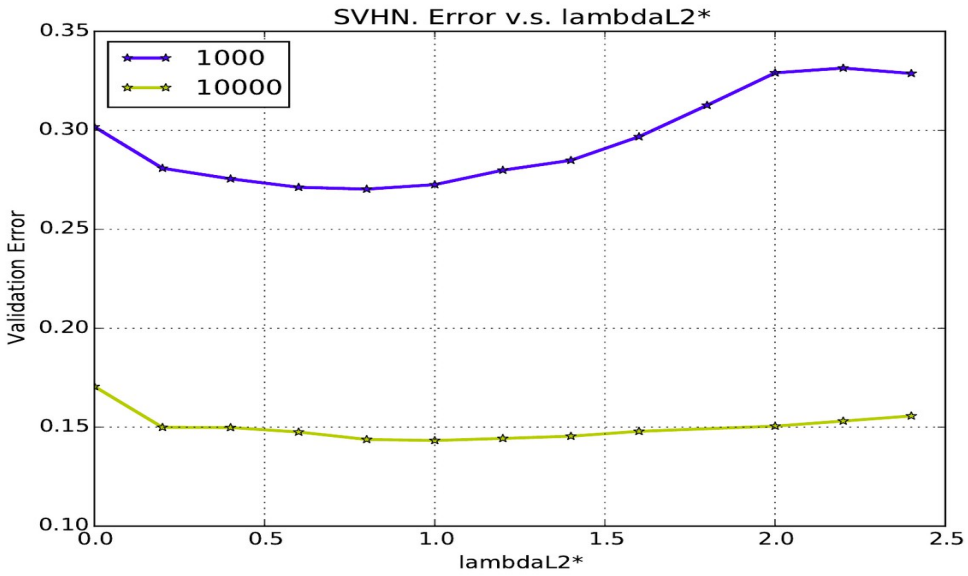


SVHN: Recognition

- **House Numbers**
- **Network:**

(128)5c-2p-(128)3c-(256)3c-2p-(256)3c-2p

- **Error rate on full dataset**
  - No reconstruction: 5.89%
  - With reconstruction: 4.94%

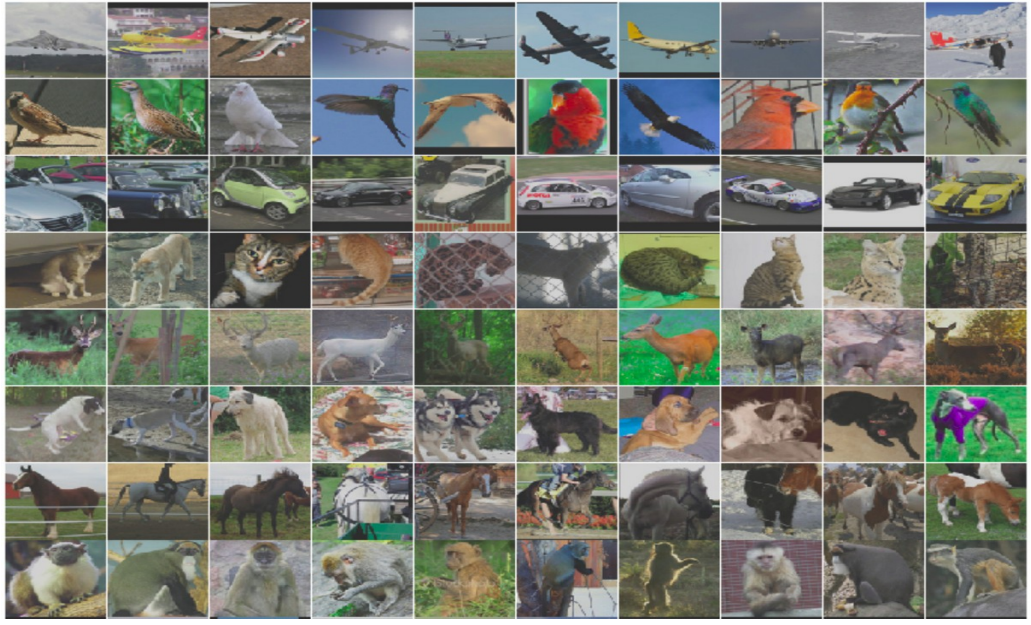


model / N	1000	10000
SWWAE	[1]27.03	[1]14.33
dp	32.25	15.72
dp-fc	[2]27.81	[2]14.39
L1	30.22	14.57
unsup-pretr	33.03	17.31
noL2M	29.12	16.51

model / N	1000 labels	error rate (in %)
KNN		77.93
TSVM		66.55
M1+KNN (Kingma et al. (2014))		65.63
M1+TSVM (Kingma et al. (2014))		54.33
M1+M2 (Kingma et al. (2014))		36.02
SWWAE without dropout ($\lambda_{L2*} = 0.8$)		27.83
SWWAE with dropout ($\lambda_{L2*} = 0.4$)		**23.56**

STL-10: Recognition

- 10 classes: airplane, bird, cat, car, deer, dog, horse, monkey, ship, truck
- 96x96 color images (from ImageNet)
- 10 predefined folds with 1000 training samples each (100 per class)
- 5000 total training samples
- 100K unlabeled samples (other categories)
- 8000 test samples (800 per class)
  [Coates et al. 2011]
- (64)3c-4p-(64)3c-3p-(128)3c-(128)3c-2p-(256)3c-(256)3c-(256)3c-(512)3c-(512)3c-(512)3c-2p-10fc

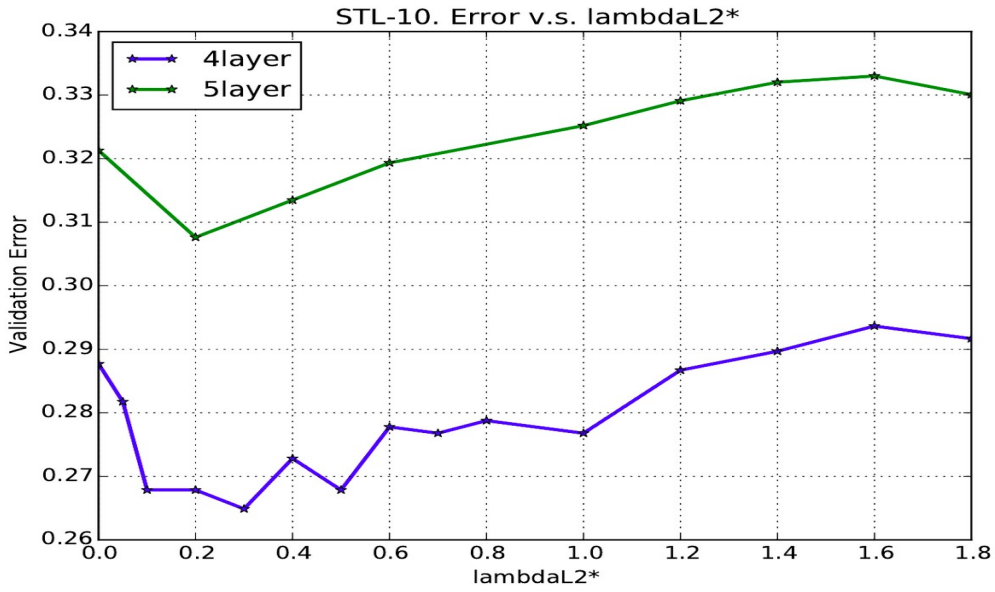


model	accuracy
Multi-task Bayesian Optimization (Swersky et al. (2013))	70.1%
Zero-bias Convnets + ADCU (Paine et al. (2014))	70.2%
Exemplar Convnets (Dosovitskiy et al. (2014))	**75.4%**
SWWAE	74.33%
Convnet of same configuration	57.45%

STL-10: Recognition

- 10 classes: airplane, bird, cat, car, deer, dog, horse, monkey, ship, truck
- 96x96 color images (from ImageNet)
- 10 predefined folds with 1000 training samples each (100 per class)
- 5000 total training samples
- 100K unlabeled samples (other categories)
- 8000 test samples (800 per class)

[Coates et al, 2011]



model	accuracy(val)
SWWAE-4layer	[1]73.51%
SWWAE-5layer	69.24%
noL2M-5layer	68.26%
noL2M-4layer	70.93%
dp-4layer	70.54%
dp-fc-4layer	71.43%

model	accuracy
Convolutional Kernel Networks [17]	62.32%
HMP [1]	64.5%
NOMP [16]	67.9%
Multi-task Bayesian Optimization [27]	70.1%
Zero-bias Convnets + ADCU [20]	70.2%
Exemplar Convnets [2]	72.8%
SWWAE-4layer	[1]74.80%

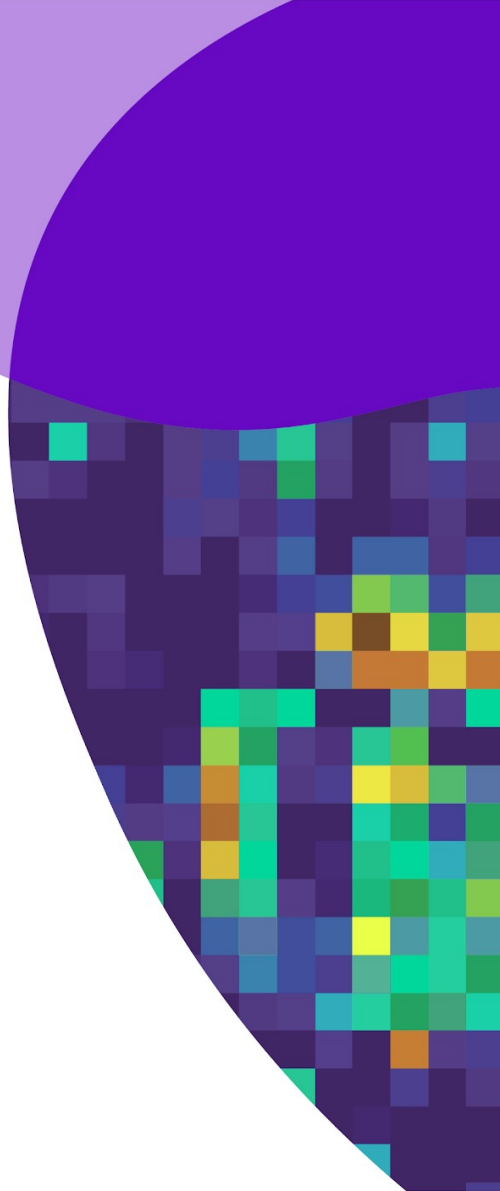
CIFAR-10, CIFAR-100, Unsup on 80M Tiny Images

- **60,000 labeled samples**
- **80 Million unlabeled samples**
- **(128)3c-(256)3c-2p-(256)3c-(512)3c-2p-(512)3c-(512)3c-2p-(512)3c-(512)3c-2p-128fc-10fc**

model	CIFAR-10	CIFAR-100
All-Convnet (Springenberg et al. (2014))	**92.75%**	66.29%
Highway Network (Srivastava et al. (2015))	92.40%	67.76%
Deeply-supervised nets (Lee et al. (2014))	92.03%	65.43%
Fractional Max-pooling with large augmentation (Graham (2014))	95.50%	68.55%
SWWAE ($\lambda_{L2rec} = 1, \lambda_{L2M} = 0.2$)	92.23%	**69.12%**
Convnet of same configuration	91.33%	67.50%

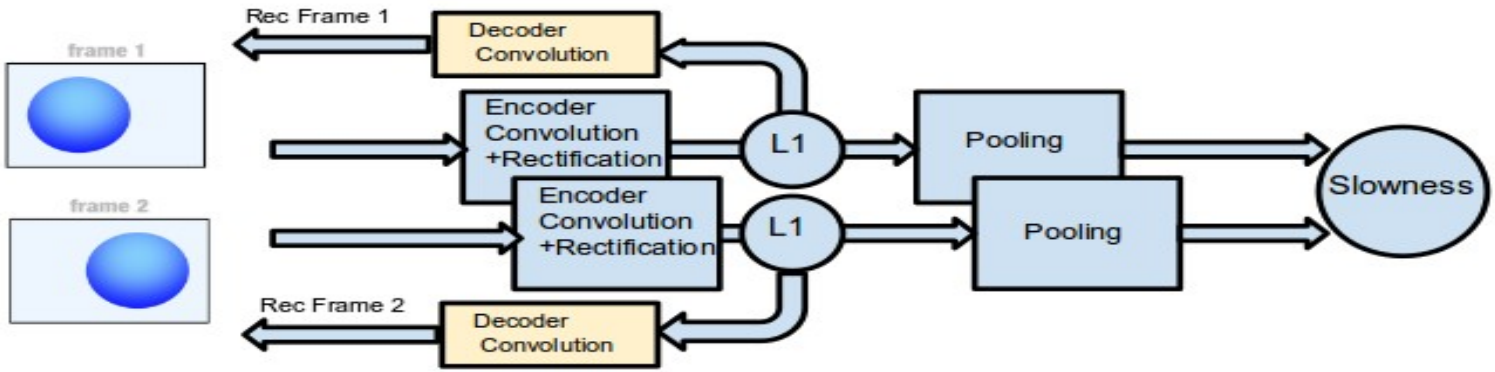
FACEBOOK AI

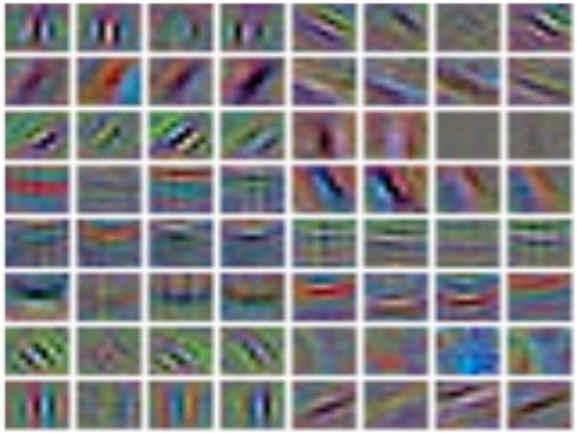
Other Latent representation regularization

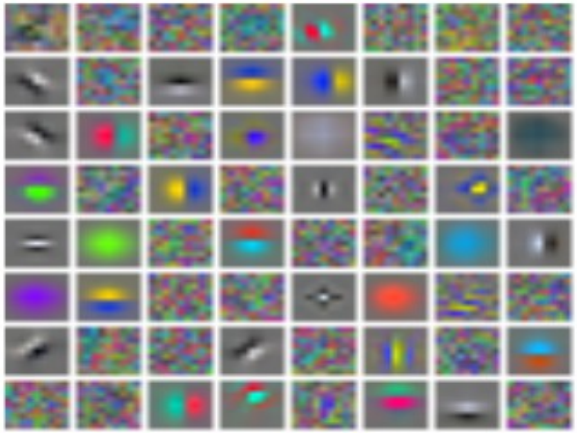


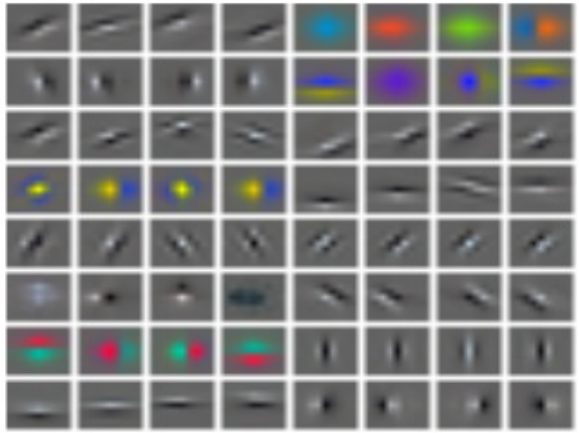
Using temporal constancy

Sparse Auto-Encoder with "Slow Feature" Penalty









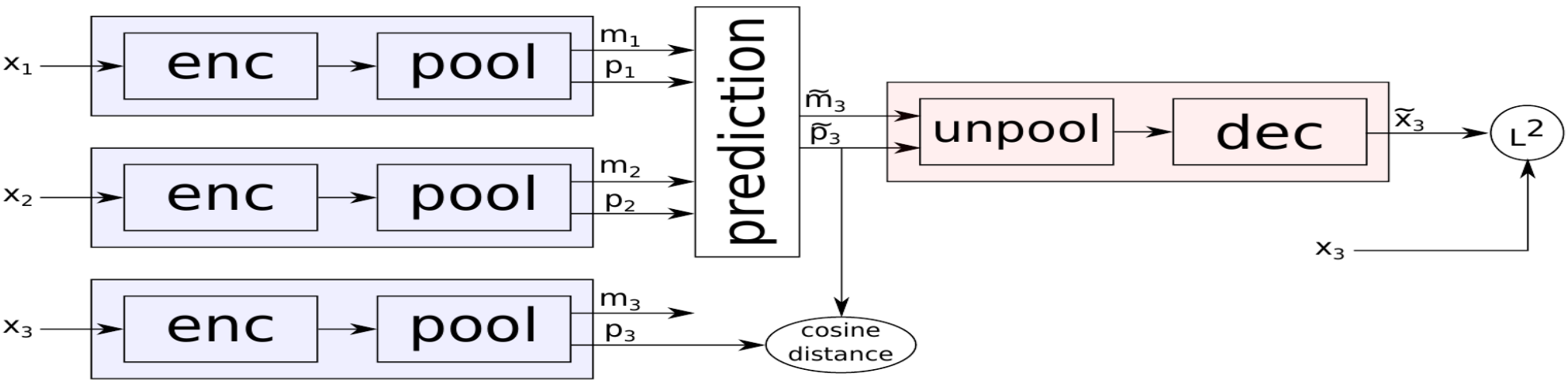
- **Supervised filters CIFAR10**
- **sparse conv. auto-encoder**
- **slow & sparse conv. auto-encoder Trained on YouTube videos**
- **[Goroshin et al. ICCV 2015, Arxiv:1412.6056]**

Unsupervised Learning by Prediction and Linearization

[Goroshin, Mathieu, LeCun NIPS 2015, arXiv:1506.03011]

- **Trained on 3 successive frames of video**
- **Maximize the colinearity between successive changes of feature vectors**
- **So that "natural" changes stay in linear manifold in feature space.**

$$L = \frac{1}{2}\|G_W(\mathbf{a}\begin{bmatrix} z^t & z^{t-1}\end{bmatrix}^T) - x^{t+1}\|_2^2 - \lambda \frac{(z^t - z^{t-1})^T(z^{t+1} - z^t)}{\|z^t - z^{t-1}\|\,\|z^{t+1} - z^t\|}$$



Unsupervised Learning by Prediction and Linearization

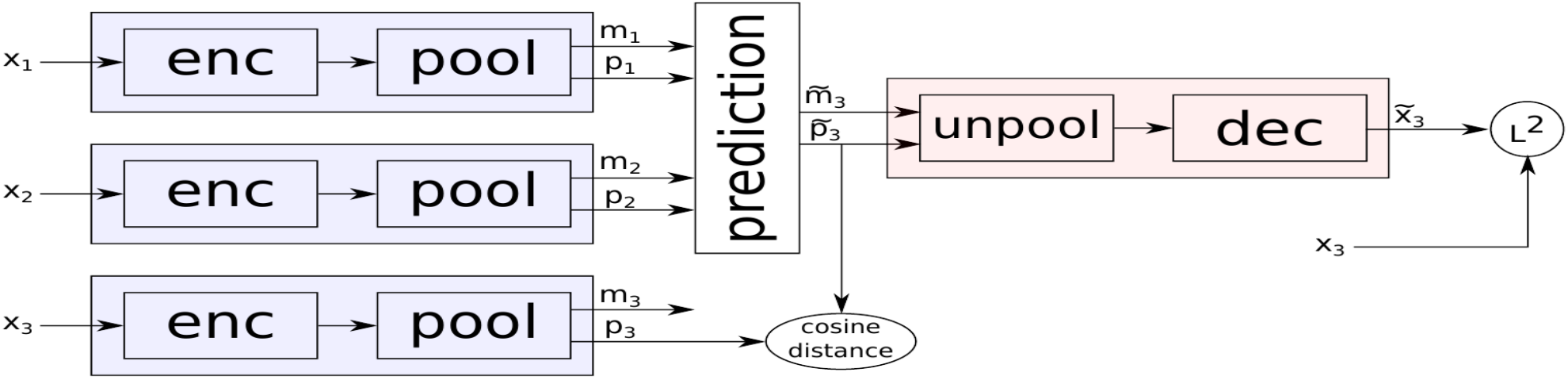
$$L = \frac{1}{2}\|G_W(\mathbf{a}\,[z^t \quad z^{t-1}]^T) - x^{t+1}\|_2^2 - \lambda \frac{(z^t - z^{t-1})^T(z^{t+1} - z^t)}{\|z^t - z^{t-1}\|\,\|z^{t+1} - z^t\|}$$

- **Magnitude**
  - (soft max)

$$m_k = \sum_{N_k} z(f,x,y)\frac{e^{\beta z(f,x,y)}}{\sum_{N_k} e^{\beta z(f',x',y')}} \approx \max_{N_k} z(f,x,y)$$

- **Phase**
  - (soft argmax)

$$\mathbf{p}_k = \sum_{N_k} \begin{bmatrix} f \\ x \\ y \end{bmatrix} \frac{e^{\beta z(f,x,y)}}{\sum_{N_k} e^{\beta z(f',x',y')}} \approx \arg\max_{N_k} z(f,x,y)$$

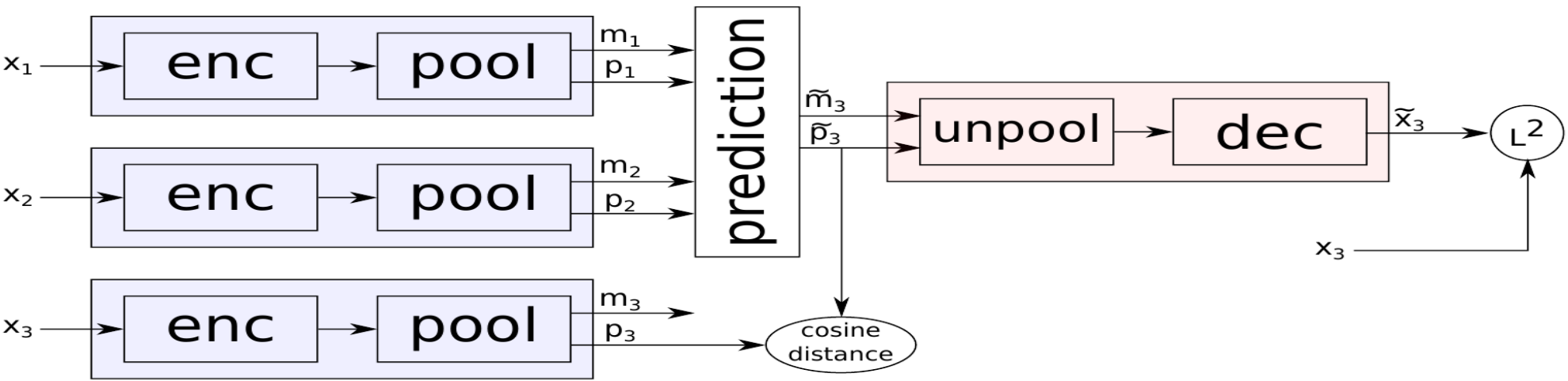


Unsupervised Learning by Prediction and Linearization

[Goroshin, Mathieu, LeCun NIPS 2015, arXiv:1506.03011]

- **Version 2:**
  - Make sure the “what”/magnitude changes slowly
  - Make sure the “where”/phase changes linearly

$$m^{t+1} = \frac{m^t + m^{t-1}}{2}$$
$$\mathbf{p}^{t+1} = 2\mathbf{p}^t - \mathbf{p}^{t-1}$$



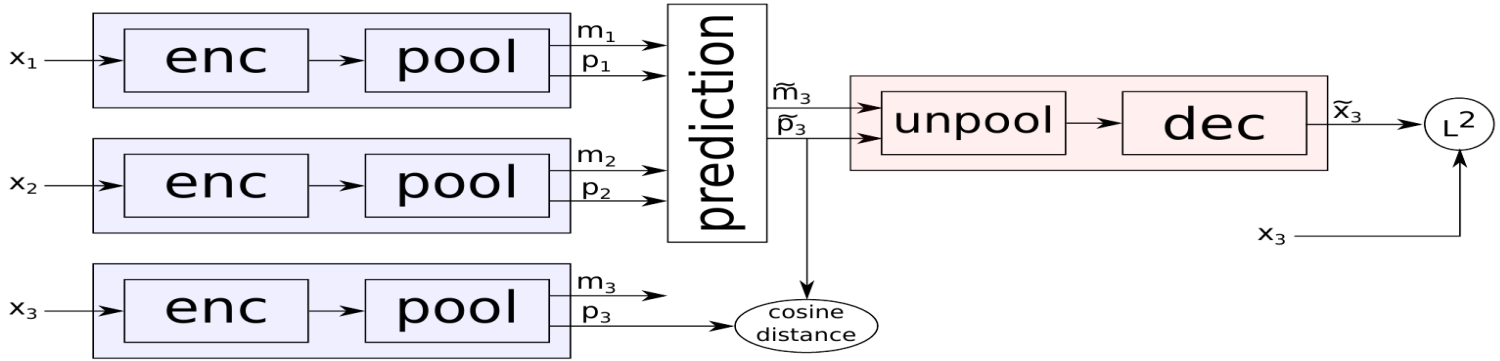
Unsupervised Learning by Prediction and Linearization

[Goroshin, Mathieu, LeCun arXiv:1506.03011]

- **Version 3: the world is unpredictable**
  - Add latent variable to compensate for that.
  - Minimize over them during training

$$\hat{z}_\delta^{t+1} = z^t + (W_1 \delta) \odot \mathbf{a} \begin{bmatrix} z^t & z^{t-1} \end{bmatrix}^T$$

$$L = \min_\delta \| G_W(\hat{z}_\delta^{t+1}) - x^{t+1} \|_2^2 - \lambda \frac{(z^t - z^{t-1})^T (z^{t+1} - z^t)}{\| z^t - z^{t-1} \| \| z^{t+1} - z^t \|}$$



Algorithm (with latent variables)

Algorithm 1 Minibatch stochastic gradient descent training for prediction with uncertainty. The number of δ-gradient descent steps (k) is treated as a hyper-parameter.

for number of training epochs **do**
  Sample a mini-batch of temporal triplets $\{x^{t-1}, x^t, x^{t+1}\}$
  Set $\delta_0 = 0$
  Forward propagate x^{t-1}, x^t through the network and obtain the codes z^{t-1}, z^t and the prediction $\hat{x}_0^{t+1}$
  for i =1 to k **do**
    Compute the L^2 prediction error
    Back propagate the error through the decoder to compute the gradient $\frac{\partial L}{\partial \delta^{i-1}}$
    Update $\delta_i = \delta_{i-1} - \alpha \frac{\partial L}{\partial \delta^{i-1}}$
    Compute $\hat{z}_{\delta_i}^{t+1} = z^t + (W_1 \delta_i) \odot \mathbf{a} \left[z^t \quad z^{t-1} \right]^T$
    Compute $\hat{x}_i^{t+1} = G_W(z_{\delta^i}^{t+1})$
  end for
  Back propagate the full encoder/predictor loss from Equation 7 using δ_k, and update the weight matrices W and W_1
end for

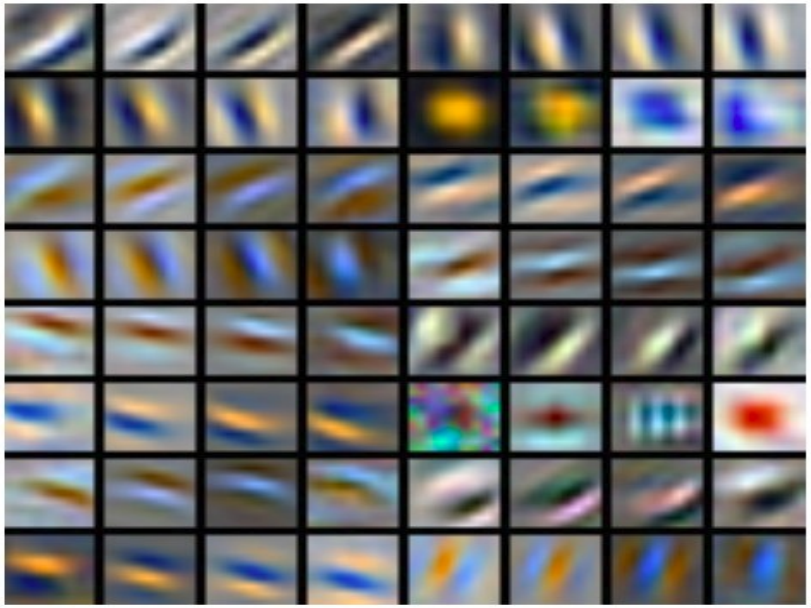
$$\hat{z}_{\delta}^{t+1} = z^t + (W_1 \delta) \odot \mathbf{a} \left[z^t \quad z^{t-1} \right]^T$$

$$L = \min_{\delta} \| G_W(\hat{z}_{\delta}^{t+1}) - x^{t+1} \|_2^2 - \lambda \frac{(z^t - z^{t-1})^T (z^{t+1} - z^t)}{\| z^t - z^{t-1} \| \| z^{t+1} - z^t \|}$$

Architectures

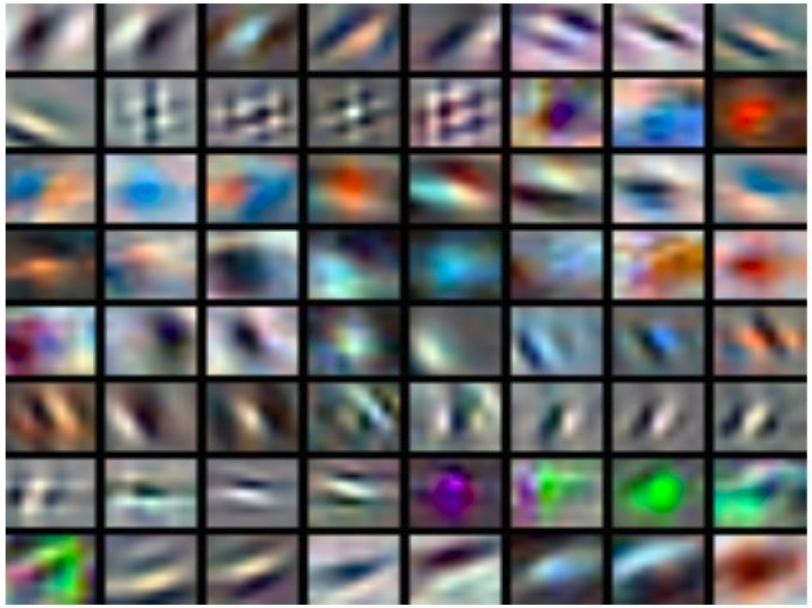
	Encoder	Prediction	Decoder
Shallow Architecture 1	Conv+ReLU $64 \times 9 \times 9$ Phase Pool 4	Average Mag. Linear Extrap. Phase	Conv $64 \times 9 \times 9$
Shallow Architecture 2	Conv+ReLU $64 \times 9 \times 9$ Phase Pool 4 stride 2	Average Mag. Linear Extrap. Phase	Conv $64 \times 9 \times 9$
Deep Architecture 1	Conv+ReLU $16 \times 9 \times 9$ Conv+ReLU $32 \times 9 \times 9$ FC+ReLU 8192×4096	None	FC+ReLU $\mathbf{8192} \times 8192$ Reshape $32 \times 16 \times 16$ SpatialPadding 8×8 Conv+ReLU $16 \times 9 \times 9$ SpatialPadding 8×8 Conv $1 \times 9 \times 9$
Deep Architecture 2	Conv+ReLU $16 \times 9 \times 9$ Conv+ReLU $32 \times 9 \times 9$ FC+ReLU 8192×4096	Linear Extrapolation	FC+ReLU 4096×8192 Reshape $32 \times 16 \times 16$ SpatialPadding 8×8 Conv+ReLU $16 \times 9 \times 9$ SpatialPadding 8×8 Conv $1 \times 9 \times 9$
Deep Architecture 3	Conv+ReLU $16 \times 9 \times 9$ Conv+ReLU $32 \times 9 \times 9$ FC+ReLU 8192×4096 Reshape $64 \times 8 \times 8$ Phase Pool 8×8	Average Mag. Linear Extrap. Phase	Unpool 8×8 FC+ReLU 4096×8192 Reshape $32 \times 16 \times 16$ SpatialPadding 8×8 Conv+ReLU $16 \times 9 \times 9$ SpatialPadding 8×8 Conv $1 \times 9 \times 9$

Filters: pooling over groups of 4 filters.



(a) Shallow Architecture 1

Non-overlapping pooling
4x4x4 (feat,x,y)

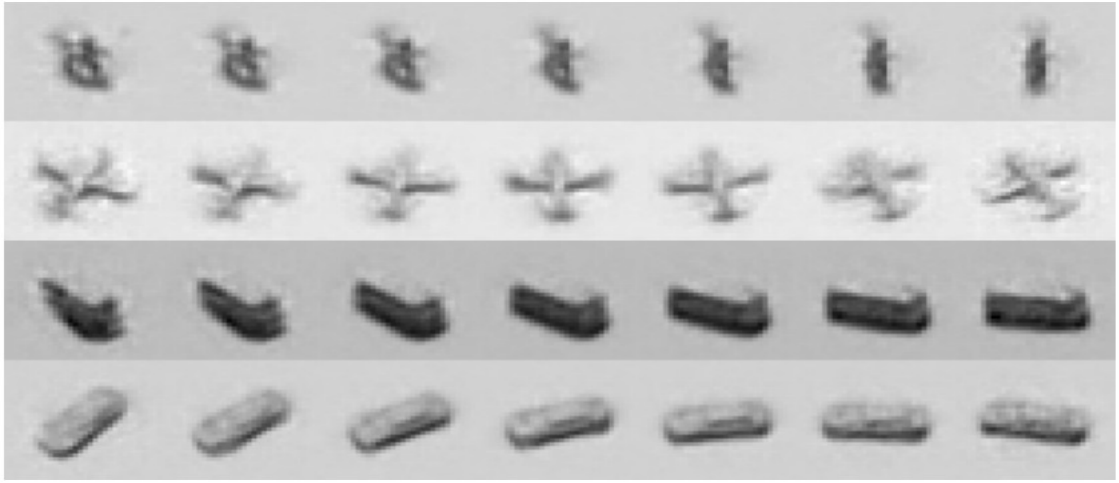


(b) Shallow Architecture 2

Overlapping pooling
4x4x4 (feat,x,y)
Stride 2 over features

Image interpolation in feature space: deterministic world

- **No phase pooling**
- **No curvature regularization**



- **With phase pooling**
- **With curvature regularization**

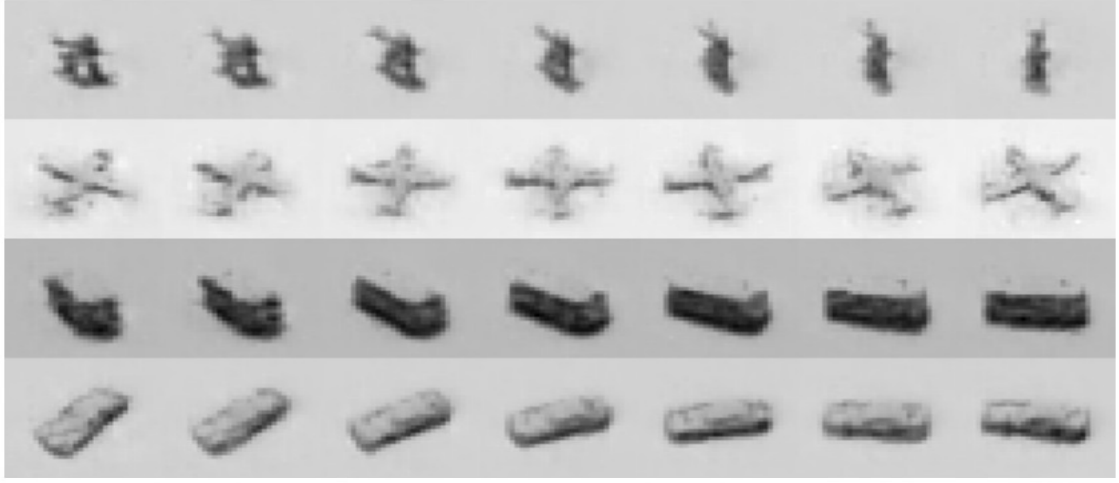


Image interpolation in feature space: deterministic world

- **Image interpolation with the basic model (siamese net)**

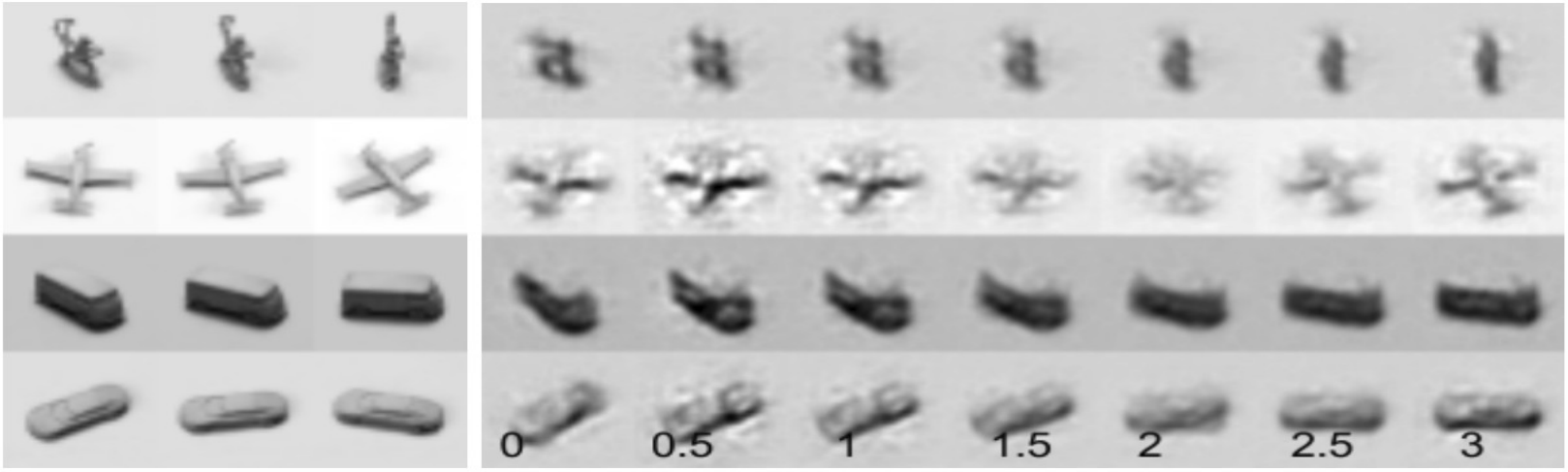
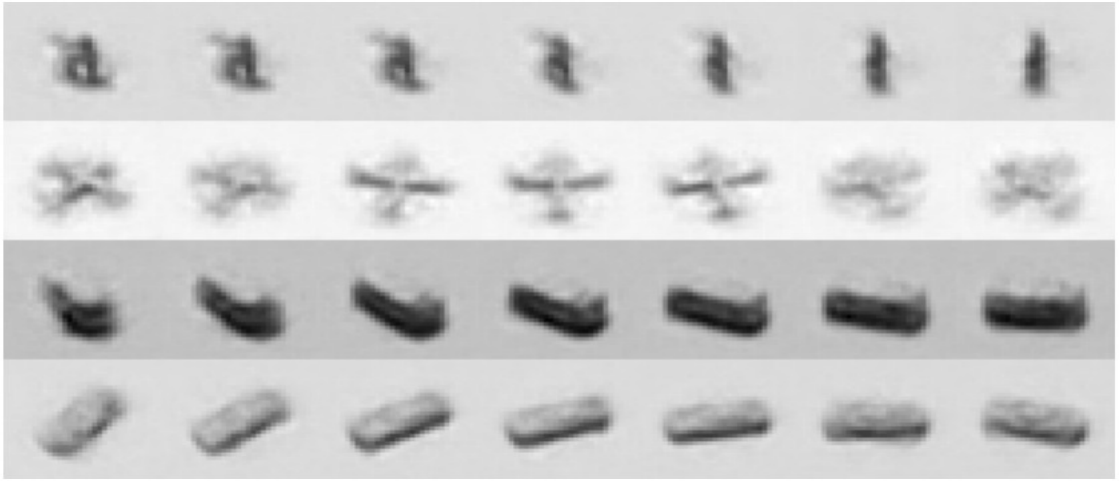


Image interpolation in feature space: unpredictable world

- **Phase interpolation**
- **No latent variable**



- **Phase interpolation**
- **With latent variable**